

Description and Use of the RSET Floating Percentile Method Spreadsheets

FPMData.xls

December 28, 2008 Revision

FPMDataGroups.xls

December 28, 2008 Revision

FPMAnova.xls

July 29, 2008 Revision

FPMCalc.xls

December 6, 2008 Revision

Including Information about Macro Codes and Forms

Michael R. Anderson
Oregon Department of Environmental Quality
Regional Sediment Evaluation Team
Freshwater Sediment Workgroup
December 30, 2008

This page intentionally left blank.

Table of Contents

1.	Introduction	3
1.1	Purpose	3
1.2	Contents	3
1.3	Spreadsheet Security	3
1.4	General Notes.....	5
2.	Spreadsheet Design and Use	6
2.1	FPMData.xls	6
2.1.1	Overview.....	6
2.1.2	Worksheets.....	6
2.1.3	Forms	7
2.1.3.1	Data Codes	7
2.1.3.2	Cell Colors.....	8
2.1.4	Instructions	8
2.2	FPMDataGroups.xls	15
2.2.1	Overview.....	15
2.2.2	Worksheets.....	15
2.2.3	Instructions	15
2.3	FPMAnova.xls.....	17
2.3.1	Overview.....	17
2.3.2	Worksheets.....	17
2.3.3	Instructions	18
2.4	FPMCCalc.xls.....	23
2.4.1	Overview.....	23
2.4.2	Worksheets.....	23
2.4.3	Forms	24
2.4.3.1	Description of Criteria Worksheet	24
2.4.3.2	Reliability Measures Definitions	24
2.4.4	Instructions	25
3.	Spreadsheet Macros	33
3.1	Introduction	33
3.1.1	Document Text Conventions	33
3.1.2	Common Naming and Macro Conventions	33
3.1.3	Macro Usage	35
3.1.3.1	The SetRanges() Macro.....	35
3.1.3.2	The RunProgram() Macro	35
3.2	Macro Code for FPMData.xls	36
3.2.1	Summary	36
3.2.1.1	Macros Activated When Spreadsheet Opens or Closes.....	36
3.2.1.2	Macros Activated from Buttons on Worksheets	36
3.2.1.3	Macros that Control the Main Function of the Program	37
3.2.2	mod01RunProgram	40
3.2.3	mod02CompileRawData	49
3.2.4	mod03SummarizeData.....	52
3.2.5	mod04MakeDataTable	56
3.2.6	mod05MakeLists	60
3.2.7	mod06FormatData	66
3.3	Macro Code for FPMDataGroups.xls	77
3.3.1	Summary	77
3.3.1.1	Macros Activated When Spreadsheet Opens or Closes.....	77
3.3.1.2	Macros Activated from Buttons on Worksheets	77
3.3.2	Module1.....	77

3.4	Macro Code for FPMAnova.xls.....	80
3.4.1	Summary	80
3.4.1.1	Macros Activated When Spreadsheet Opens or Closes.....	80
3.4.1.2	Macros Activated from Buttons on Worksheets	80
3.4.1.3	Macros that Control the Main Function of the Program	81
3.4.2	mod01RunProgram	84
3.4.3	mod02Anova	93
3.4.4	mod03Sort Data	95
3.4.5	mod04Format Pages.....	96
3.5	Macro Code for FPMCalc.xls.....	101
3.5.1	Summary	101
3.5.1.1	Macros Activated When Spreadsheet Opens or Closes.....	101
3.5.1.2	Macros Activated from Buttons on Worksheets	101
3.5.1.3	Macros that Control the Main Function of the Program	102
3.5.2	mod01RunProgram	106
3.5.3	mod02CreateDistributions	116
3.5.4	mod03SummarizeData	117
3.5.5	mod04CalculatePercentiles.....	119
3.5.6	mod05ErrorCalculations	120
3.5.7	mod06SelectCriteria	122
3.5.8	mod07FloatingPercentilePass1	124
3.5.9	mod08FloatingPercentilePass2.....	128
3.5.10	mod09CountCopyData	132
3.5.11	mod10FormatData.....	139

1. Introduction

1.1 Purpose

The purpose of this document is to explain how the Regional Sediment Evaluation Team (RSET) Floating Percentile Method (FPM) Microsoft Excel® Spreadsheets work so that they may be more easily used by data analysts as well as revised and updated if changes are required in the future.

The Floating Percentile Method was developed by Dr. Teresa Michelsen of Avocet Consulting and uses paired sediment toxicity and chemical analytical data to develop sediment quality guidelines. The spreadsheets were developed by Michael R. Anderson of the Oregon Department of Environmental Quality as one of the tasks of the RSET Freshwater Sediment Workgroup.

This document was written under the assumption that the people who use it are at least moderately experienced in Microsoft Excel® and are familiar with the RSET FPM.

1.2 Contents

This document describes the worksheets and forms for the following spreadsheets:

FPMDData.xls (Revised December 28, 2008);

FPMDDataGroups.xls (Revised December 28, 2008);

FPMAAnova.xls (Revised July 29, 2008); and

FPMCalc.xls (Revised December 6, 2008).

It also includes the code for all of the macros used to perform the spreadsheet calculations and provides a brief explanation of the macros. It is assumed, however, that readers who refer to the macro code will have a basic understanding of Microsoft's Visual Basic Language for Applications. Therefore, detailed explanations are not provided for all of the code.

Some of the macros used in these documents were originally adapted from examples in *Excel for Windows 95 Power Programming with VBA* (IDG Books Worldwide, 1996) by John Walkenbach. Newer versions of this book are available. Mr. Walkenbach also maintains a helpful website – The Spreadsheet Page – that was referred to on occasion and which can be found at <http://j-walk.com/ss/>.

1.3 Spreadsheet Security

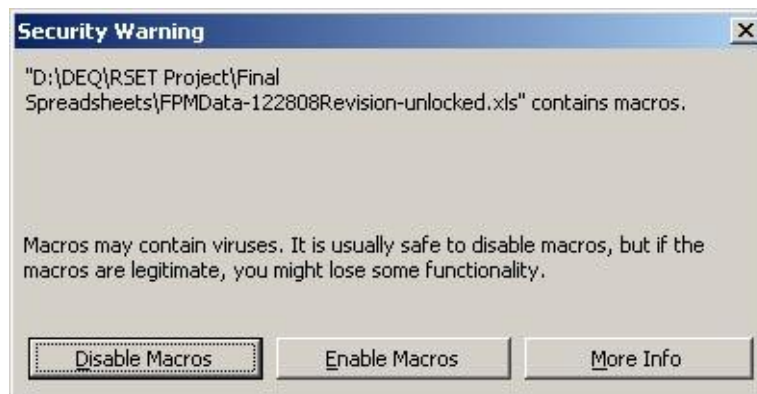
All of the FPM spreadsheets contain macros, which are used to perform the necessary calculations, sorting, and other manipulation of the data. If the security level of your copy of Microsoft Excel® is set too high, the macros will be disabled by the software. To check the security level of your software, click on the "Tools" menu and select "Options" from the bottom of that menu.

Click on the "Security" tab at the top of the Options menu and then click on the "Macro Security..." button on the lower right side of that page. You should then see the "Security" menu shown on the top of the next page.

Select "Medium," click OK to save the selection and close the "Security" menu, and OK to close the "Options page." This will allow you to open the FPM spreadsheets.



When you open any of the spreadsheets you should get the security warning shown below. For the spreadsheet to function you must select “Enable Macros” to continue. However, if you only need to look at something in the spreadsheet, it will open and close more quickly if you select “Disable Macros.”



The macros that run the spreadsheets are protected, which means that a password is required before you can view or edit them. However, all of the FPM spreadsheets are unprotected and the cells in the spreadsheets are unlocked. This was done to allow users the flexibility they might need to sort and rearrange data in ways other than those programmed into the spreadsheets. However, this also means that it is very easy to accidentally delete or paste over formulas that are written in some of the cells. Loss of such formulas will, of course, prevent the spreadsheets from generating valid results. Therefore, any changes that you make to the spreadsheets other than adding data to the designated input tables and following the provided instructions must be done with due caution. Always keep a backup copy of the original spreadsheets.

1.4 General Notes

When you open any of the FPM Spreadsheets you will notice that some of the menus at the top of the spreadsheet are hidden from view. This is done to provide you with the maximum amount of useful space on your screen and reduce the amount of scrolling that you might have to do. If you want to restore any of the menus, open the "View" menu and scroll down to "Toolbars." When the Toolbars menu appears click on "Standard." Repeat this step and click on "Formatting." These are the two toolbars that are typically showing when Excel opens up. If you want to know if a certain cell has a formula in it you should open the "View" menu and click on "Formula Bar." When you select a cell you can then see if it contains a number or a formula. As noted in the previous section, it is very important that you not accidentally delete or paste over formulas.

If you decide to paste data into any of the spreadsheets it is recommended that you use the "Paste Special" command instead of the usual "Paste" command. After you have copied the data that you want to paste and have highlighted the area where you want to paste it, open the "Edit" menu and then click on "Paste Special." When the Paste Special dialog box opens, click on the "Values" selection in the top half of the box and then click on OK to close the box and paste the data. Using this method you will not accidentally reformat any of the cells in the spreadsheet or paste formulas into the cells instead of numbers.

Each of the spreadsheets has options that affect that way the spreadsheet functions. You will need to indicate how you want the program to handle these options before you can successfully run the spreadsheet macros. The cells in which you designate your choices are shown as white cells. Cells that are yellow are filled in by the macros. The exceptions to this are the large data tables, all of which are white. For example, FPMCalc.xls has the following table on the [ControlScreen] page. You will need to supply values for Initial False Negative Target, Final False Negative Target, Target Interval, Number of Increments, and Percent Precision. You also indicate if you want to Pre-Screen AETs, Omit AET Outliers, and designate how you want to define an outlier. The macros will fill in the Number of Stations, Number of Hits, etc.

Data Summary and Program Settings	
Chemical Data Summary	
Number of Analytes	46
No. of Data Points	10890
Number of Stations	565
Number of Samples	568
Number of Hits	67
Number of Indeterminates	0
Number of No-Hits	501
Criteria Table Settings	
Initial False Negative Target	5
Final False Negative Target	25
Target Interval	5
Number of Steps	5
Number of Increments	10
Percent Precision	10
AET and Outlier Settings	
Pre-Screen AETs (Y or N)?	N
Omit AET Outliers (Y or N)?	N
Outlier exceeds by multiple	3

Regarding nomenclature, the words "page" and "worksheet" are used interchangeably as are "spreadsheet" and "workbook." Names shown on the tab of a page/worksheet are referred to in [SquareBrackets]. The FPM spreadsheets were created with the PC Edition of Microsoft Excel® 2003 and may not work properly with other versions of Excel, including Excel for Macs, or other spreadsheets; however, a 2007 version of the ANOVA spreadsheet has been created for use with the Office 2007 analytical toolpak.

2. Spreadsheet Design and Use

2.1 *FPMData.xls*

2.1.1 Overview

FPMData is the first of the three spreadsheets used for implementing the FPM. Its primary purpose is to help you sort and screen large sets of analytical data from sediment samples to generate a more concise data set that meets requirements that you specify. This spreadsheet can:

1. Accept up to 65,000 analytical results for up to 250 chemical constituents having up to 100 different data qualifiers,
2. Sort the raw data into groups by chemical constituent,
3. Summarize the data for the data set as a whole and for each constituent, and
4. Generate a final data set with options to:
5. Screen out data points with specified data qualifiers,
6. Screen out constituents with less than a specified number of data,
7. Delete specified constituents by name, and
8. Create up to 25 new analyte groups and sum the data for any number of individual constituents.

2.1.2 Worksheets

This spreadsheet workbook consists of 7 worksheets (pages):

“InputData” is the page on which you store all of the initial analytical data. You must place the data into the “Initial Analytical Data” table in the exact order defined by the existing column headings: *Survey, Station, Sample, Chemical, Conc, Units, and Quality*. Instructions are provided on the worksheet.

“OmitAnalytes” is the page you can use to generate tables listing all available analytes and all data qualifier codes found in the initial analytical data. After generating the tables you can select analytes and data qualifier codes to be omitted from all subsequent lists and calculations by putting a check mark in the box to the right of the name. Instructions are provided on the worksheet.

“CreateSums” is the page where you can define new groups of analytes from the existing individual analytes. After you define a group and give it a name, the spreadsheet will sum the relevant individual analytical data into a total value for that group. You can also save your groups for future use. Instructions are provided on the worksheet.

“AnalyteSummary” contains a table which lists all of the initial analytical data along with data for newly defined groups. Color codes are used to identify data points or analytes that have been omitted or summed as defined in the previous two worksheets.

“DataTable” lists the final edited version of the initial data set which omits all relevant data points and analytes selected on the “OmitAnalytes” page and includes the new groups defined on the “CreateSums” page.

"DataSetSummary" lists facts about the data set such as total number of data points, how many were omitted, how many were summed, *etc.* and provides a table showing all of the new groups and the members of those groups as well as the names of any analytes screened out from the initial data set. There is an option on this page that you can use to save a set of designated groups and group members for future use in a different FPMData spreadsheet, or recover and reuse a previously saved set of groups and group members. Instructions for doing this are provided on the worksheet.

"Names" is a reference worksheet that lists all of the named cells used in all of the worksheets. It also contains a table used to store the names of the tool bars that are open when the program begins. The colors used in the summary data table are listed there for reference. This is normally a hidden page.

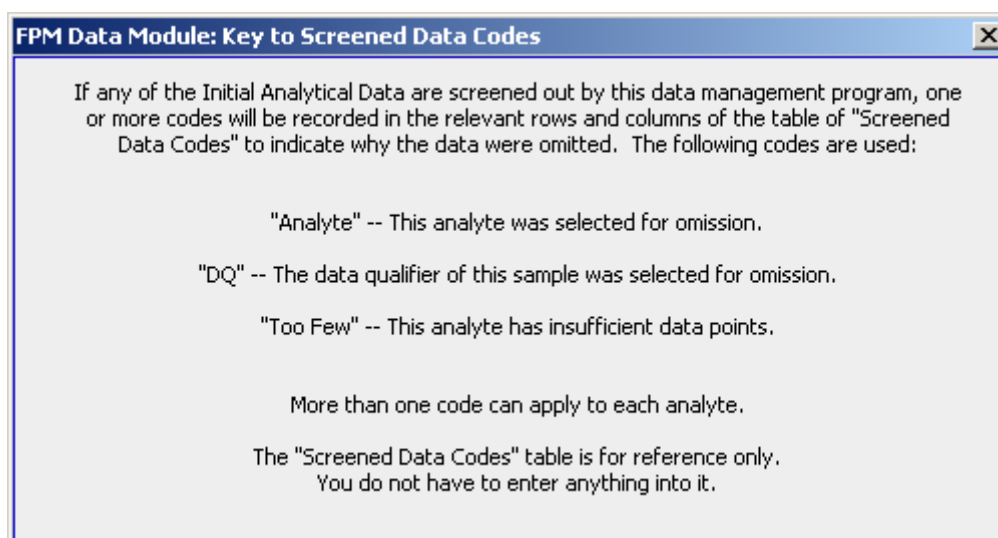
This workbook also contains 2 forms (**Key to Screened Data Codes** and **Key to Raw Data Screen Cell Colors**) that provide reference information.

Finally, this workbook includes 6 modules that contain the main macro code for running the entire workbook. The forms and the macro modules are password-protected; however, none of the worksheets in the workbook are protected so you must be careful not to paste data into cells that may already contain formulas. If you follow the instructions on each worksheet page, that should not be a problem.

2.1.3 Forms

2.1.3.1 Data Codes

This form is opened by a button on the "InputData" page. It defines the three codes used in the "Screened Data Codes" table.



FPM Data Module: Key to Screened Data Codes

If any of the Initial Analytical Data are screened out by this data management program, one or more codes will be recorded in the relevant rows and columns of the table of "Screened Data Codes" to indicate why the data were omitted. The following codes are used:

"Analyte" -- This analyte was selected for omission.

"DQ" -- The data qualifier of this sample was selected for omission.

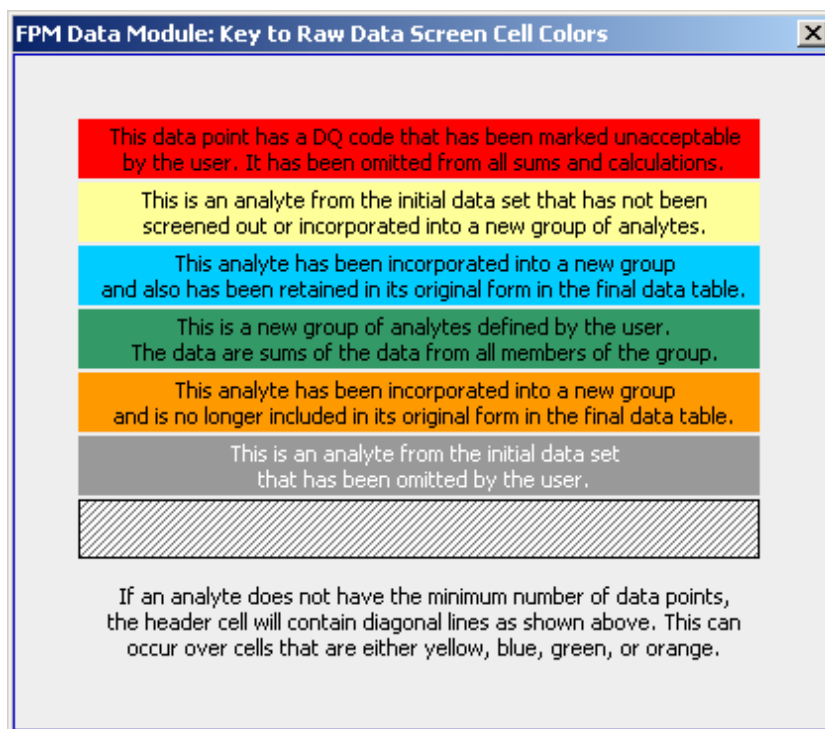
"Too Few" -- This analyte has insufficient data points.

More than one code can apply to each analyte.

The "Screened Data Codes" table is for reference only.
You do not have to enter anything into it.

2.1.3.2 Cell Colors

This form is opened by a button on the “AnalyteSummary” page. It explains the various color codes used in the “Full Data Set Sorted by Analyte” table.



2.1.4 Instructions

Instructions for running this program along with descriptions of images from the FPMData spreadsheet are given below. Instructions are also provided on the [InputData], [OmitAnalytes], [CreateSums], and [DataSetSummary] worksheets in the body of the spreadsheet.

As noted in the section on Spreadsheet Security, when you open the spreadsheet you will see a warning that the spreadsheet contains macros. Be sure to click on the “Enable Macros” button.

Before you place new data into the FPMData spreadsheet, use the “Clear Analytical Data” and “Clear Previous” buttons on the [InputData] worksheet (shown below) to make sure that all data left over from a previous data set have been removed. Then place your raw analytical data, including the relevant identifiers, into the “Initial Analytical Data” table on the [InputData] worksheet. The data must be placed into the columns in the order shown – Survey, Station, Sample, Chemical, Concentration, Units, and Data Quality. Do not leave any empty rows in the middle of the data set and do not change the order of the columns. If you are not going to enter new data, but plan to use the data that are already in this table, click on the “Clear Previous” button to erase any remaining results from a previous use without deleting the data that are in the “Input Analytical Data” table.

The data storage table on the [InputData] worksheet in the FPMData.xls spreadsheet

[illegible]

Indicate the lowest number of data points that you consider acceptable for the analytes that you are testing by putting an integer into the cell next to "Omit Chem w/Data Pts <". For the analyses carried out by the RSET team, 30 was used as the minimum number.

Data Preparation for Floating Percentile Calculations

Page 2: Omit Selected Analytes and Data Qualifiers

Generate Data Lists

Instructions for this Page	Analyte List	Units	Omit	Data Qualifier List	Omit
<p>After the analytical data have been entered into the [InputData] worksheet, use the Generate Data Lists button to populate the Analyte List and Data Qualifier List for this data set.</p> <p>To OMIT selected data from the final data table, do one or both of the following:</p> <p>(1) To omit specific analytes, check the box(es) next to the relevant name(s) in the Analyte List.</p> <p>(2) To omit analytes with specific data qualifiers, check the box(es) next to the appropriate symbol(s) in the Data Qualifier List.</p> <p>In the box labeled Minimum No. of Data Points, enter the number below which an analyte will be excluded from the calculations. This box must contain an integer greater than or equal to 1 in order for the program to operate.</p> <p>Go to the [CreateSums] worksheet.</p>	1,2,3-Trichloropropane	PPB	<input type="checkbox"/>	B	<input type="checkbox"/>
	1,2,4-Trichlorobenzene	PPB	<input type="checkbox"/>	BEJK	<input type="checkbox"/>
	1,2-Dichlorobenzene	PPB	<input type="checkbox"/>	BJE	<input type="checkbox"/>
	1,2-Dichloroethane	PPB	<input type="checkbox"/>	BJK	<input type="checkbox"/>
	1,4-Dichlorobenzene	PPB	<input type="checkbox"/>	BJT	<input type="checkbox"/>
	1-Methylnaphthalene	PPB	<input type="checkbox"/>	BNJK	<input type="checkbox"/>
	2,3,4,5-Tetrachlorophenol	PPB	<input type="checkbox"/>	BNTJK	<input type="checkbox"/>
	2,4,5-Trichlorophenol	PPB	<input type="checkbox"/>	BTNJK	<input type="checkbox"/>
	2,4,6-Trichlorophenol	PPB	<input type="checkbox"/>	D	<input type="checkbox"/>
	2,4-D	PPB	<input type="checkbox"/>	J	<input type="checkbox"/>
	2,4-DB	PPB	<input type="checkbox"/>	JB	<input type="checkbox"/>
	2,4'-DDD	PPB	<input type="checkbox"/>	JBEL	<input type="checkbox"/>
	2,4'-DDE	PPB	<input type="checkbox"/>	JBG	<input type="checkbox"/>
	2,4'-DDT	PPB	<input type="checkbox"/>	JBK	<input type="checkbox"/>
	2,4-Dichlorophenol	PPB	<input type="checkbox"/>	JBL	<input type="checkbox"/>
	2,4-Dimethylphenol	PPB	<input type="checkbox"/>	JBT	<input type="checkbox"/>
	2,4-Dinitrotoluene	PPB	<input type="checkbox"/>	JBTG	<input type="checkbox"/>
	2-Chlorophenol	PPB	<input type="checkbox"/>	JE	<input type="checkbox"/>
	2-Methylnaphthalene	PPB	<input type="checkbox"/>	JEL	<input type="checkbox"/>
	2-Methylphenol	PPB	<input type="checkbox"/>	JG	<input type="checkbox"/>
	4,4'-DDD	PPB	<input type="checkbox"/>	JGB	<input type="checkbox"/>
	4,4'-DDE	PPB	<input type="checkbox"/>	JK	<input type="checkbox"/>
	4,4'-DDT	PPB	<input type="checkbox"/>	JKB	<input type="checkbox"/>
	4-Chloro-3-methylphenol	PPB	<input type="checkbox"/>	JKE	<input type="checkbox"/>
	4-Methylphenol	PPB	<input type="checkbox"/>	JL	<input type="checkbox"/>
4-Nitroaniline	PPB	<input type="checkbox"/>	JP	<input type="checkbox"/>	
Acenaphthene	PPB	<input type="checkbox"/>	JPD	<input type="checkbox"/>	
Acenaphthylene	PPB	<input type="checkbox"/>	JT	<input checked="" type="checkbox"/>	
Acetone	PPB	<input type="checkbox"/>	JTB	<input type="checkbox"/>	

The [OmitAnalytes] worksheet in the FPMData.xls spreadsheet

Open the [CreateSums] worksheet (shown below) and click on the "Create Analyte List" button at the top of the page. This will generate a list of analytes similar to the one previously generated on the [OmitAnalytes] worksheet. If you put check marks on the [OmitAnalytes] worksheet to omit certain analytes, those analyte names will be shown on the [CreateSums] page in dark gray cells.

Data Preparation for Floating Percentile Calculations

Page 3: Define Summed Analyte Groups

Create Analyte List

Retain Group List

Clear Lists

Create Data Table

Instructions for this Page	Group Names List	Units	Group No.	Analyte List	Units	Group No.	Retain
<p>1. After completing the OmitAnalytes page, you have several options for working with groups of analytes.</p> <p>a. To save the current group list or recover a previously saved group list, see the instructions on the [DataSetSummary] page.</p> <p>b. To revise the current groups for the existing set of analytical data, see the instructions at the bottom of the next column.</p> <p>c. If you entered a new data set on the [InputData] page or made any changes on the [OmitAnalytes] page, continue with step 2.</p>	DDTs	PPB	1	1,2,3-Trichloropropane	PPB		<input type="checkbox"/>
	Aroclors	PPB	2	1,2,4-Trichlorobenzene	PPB		<input type="checkbox"/>
			3	1,2-Dichlorobenzene	PPB		<input type="checkbox"/>
			4	1,2-Dichloroethane	PPB		<input type="checkbox"/>
			5	1,4-Dichlorobenzene	PPB		<input type="checkbox"/>
			6	1-Methylnaphthalene	PPB		<input type="checkbox"/>
			7	2,3,4,5-Tetrachlorophenol	PPB		<input type="checkbox"/>
			8	2,4,5-Trichlorophenol	PPB		<input type="checkbox"/>
			9	2,4,6-Trichlorophenol	PPB		<input type="checkbox"/>
			10	2,4-D	PPB		<input type="checkbox"/>
			11	2,4-DB	PPB		<input type="checkbox"/>
			12	2,4'-DDD	PPB	1	<input type="checkbox"/>
			13	2,4'-DDE	PPB	1	<input checked="" type="checkbox"/>
			14	2,4'-DDT	PPB	1	<input type="checkbox"/>

The [CreateSums] worksheet in the FPMData.xls spreadsheet

To create a group that consists of the sum of two or more listed analytes, type a name for that group into the first available line in the "Group Names List." Note the "Group No." associated with the line containing your new group

name. Type that number in the “Group No.” column of the “Analyte List” in the cell next to every analyte that is a part of the new group.

For example, if you wanted to sum all of the DDT-related compounds, you could create a group named DDTs (see above). After putting that name in the first line of the “Group Names List,” you would put the number 1 in the “Group No.” column next to each compound you wanted to include in the DDT group. If you also wanted to group the Aroclors, you could put that name into the second line of group names and type the number 2 next to each compound you wanted to include in the Aroclors group.

When analytes are made part of a new group, the data for the individual analytes normally will not appear in the final screened data table. If you decide that you want to keep one or more of these individual analytes in addition to adding it to a group, put a check mark in the “Retain” column next to an analyte that you wish to retain.

After you have created a set of groups, there are several ways that you can save and reuse them:

1. If you would like to save the groups for continued reuse in the spreadsheet where you created them, put a check mark in the “Retain Group List” box at the top of the page. This will prevent the group information from being deleted when you click on the “Clear Previous” button on the [InputData] page.
2. If you would like to save the groups for use in other FPMDData spreadsheets, enter a name for the group in the appropriate cell in the Save Current Group box, then enter the name of the file into which the group data will be stored. Make sure this file is open in the same folder as your spreadsheet, then click on the Save Group button.

Save Current Group
Enter a name for the group below.
Complete DDT and Aroclor
Enter the name of the storage file below.
FPMDDataGroups.xls
Save Group

If you would like to reuse a previously saved set of groups instead of defining new ones, enter the name for the desired groups in the appropriate cell in the Retrieve Saved Group box, then enter the name of the file in which the group data is stored. Make sure this file is open in the same folder as your spreadsheet, then click on the Retrieve Group button.

Retrieve Saved Group
Enter the name of the group below.
Partial DDT and Aroclor
Enter the name of the storage file below.
FPMDDataGroups.xls
Retrieve Group

The default name for the spreadsheet in which group information is stored or recovered is FPMDDataGroups.xls. The file does not have to have this name, but it must be a copy of the original FPMDDataGroups.xls, which will be discussed in the next section.

Click on the “Create Data Table” button to carry out the requested sorting, screening, and grouping. When the program starts it will check for certain errors or omissions. If one is encountered the program will terminate and a message box will appear on the screen telling you what is wrong (for example, if the analytes in a group do not all have the same units). When the calculations are complete a message indicating the successful completion will appear on the computer screen.

The results will show up on two worksheets, [AnalyteSummary] and [DataTable]. A table at the top of the [AnalyteSummary] worksheet (shown below) summarizes data about each analyte such as the number of data points, the maximum concentration, the minimum concentration, etc.

Data Preparation for Floating Percentile Calculations							
Page 4: Summary of Analytes							
Click Here to View Key to Cell Colors							
Property	2-Methylnaphthalene	4-Methylphenol	Acenaphthene	Acenaphthylene	Acetone	Aldrin	alpha-Endosulf
Total Number of Data Points	308	92	308	308	81	72	52
Number with Unacceptable Data Qualifiers	36	21	25	30	0	1	0
Number Used in Final Data Set	272	71	283	278	81	71	52
Minimum concentration	0	4	0	0	3	0	0
Maximum concentration	37000	510	89000	11000	42	691	21.1

The data point and min/max concentration summaries on the [AnalyteSummary] worksheet in the FPMData.xls spreadsheet

The [AnalyteSummary] worksheet also contains a table of the sample concentrations listed by analyte (shown below), both the ones from the raw data table and any new groups that you defined. The cells that contain the analyte names can be any of several colors. The light yellow cells shown at the beginning of the table contain analytes from your raw data list that were not screened out. Individual cells shown in red identify data having data qualifiers that you chose to omit. Those data points will not show up in the final screened data table.

Full Data Set Sorted by Analyte								
Survey	Station	Sample	2-Methylnaphthalene	4-Methylphenol	Acenaphthene	Acenaphthylene	Acetone	Aldrin
			PPB	PPB	PPB	PPB	PPB	PPB
FWPHBR04	FWPHBR-D1-1	D1-1	1.7	6.6	1.4	3.4		
FWPHBR04	FWPHBR-D2	D2	2.6		4.6	13		
FWPHBR04	FWPHBR-G007	G007-1	3.2		5.7	3.5		
FWPHBR04	FWPHBR-G009	G009	5.5		8.1	9.7		
FWPHBR04	FWPHBR-G010	G010	4.9		4.1	6.7		
FWPHBR04	FWPHBR-G011	G011	7.5	5.5	8.8	16		
FWPHBR04	FWPHBR-G015	G015	9.6	8.8	9.7	36		
FWPHBR04	FWPHBR-G017	G017	8.5	16	8.2	19		
FWPHBR04	FWPHBR-G019	G019	6.8		4.9	12	0.902	
FWPHBR04	FWPHBR-G020	G020	10	9.7	18	10		
FWPHBR04	FWPHBR-G024	G024	54		98	12	1.38	
FWPHBR04	FWPHBR-G025	G025	7.9		13	11		
FWPHBR04	FWPHBR-G026	G026	3.3	15	3.4	4.5		
FWPHBR04	FWPHBR-G027	G027	1.3		0.86	1.6		
FWPHBR04	FWPHBR-G033	G033	5.6		360	4.7		

The sample name and concentration lists on the [AnalyteSummary] worksheet in the FPMData.xls spreadsheet

As you scroll to the right in this table you will encounter colors other than light yellow (shown below). These colors denote analytes that you have omitted, new groups that you created, etc. The key to the colors can be found by clicking on the "Click Here to View Key to Cell Colors" button at the top of the worksheet page or by looking in Section 2.1.3 in this guidance document.

Full Data Set Sorted by Analyte											
Survey	Station	Sample	2,4'-DDT	DDTs	Aroclors	2,4'-DDD	2,4'-DDE	4,4'-DDD	4,4'-DDE	4,4'-DDT	Aroclor
			PPB	PPB	PPB	PPB	PPB	PPB	PPB	PPB	PPE
FWPHBR04	FWPHBR-D1-1	D1-1	0.336	5.873	20.93	0.344	0.727	1.84	2.37	0.256	4.1
FWPHBR04	FWPHBR-D2	D2	0.194	2.6858	11.73	0.191	0.0218	1.28	0.83	0.169	
FWPHBR04	FWPHBR-G007-1	G007-1	0.037	5.263	293	0.711	0.028	1.31	2.99	0.187	230
FWPHBR04	FWPHBR-G009	G009	3.38	39.2686	1760	29.7	0.0196	2.06	3.11	0.999	140
FWPHBR04	FWPHBR-G010	G010	0.854	9.33515	168	0.03015	0.369	5.44	0.722	1.92	120
FWPHBR04	FWPHBR-G011	G011	1.99	22.573	1660	10.9	0.023	3.96	4.24	1.46	130
FWPHBR04	FWPHBR-G015	G015	2.84	14.3437	1750	2.45	0.0237	2.88	2.98	3.17	140
FWPHBR04	FWPHBR-G017	G017	1.83	10.0494	750	0.561	0.0284	3.18	2.97	1.48	550
FWPHBR04	FWPHBR-G019	G019	1.96	6.69835	1530	0.517	0.01635	1.65	0.835	1.72	800
FWPHBR04	FWPHBR-G020	G020	1.13	8.1296	440	1.57	0.0296	1.71	2.61	1.08	360
FWPHBR04	FWPHBR-G024	G024	0.467	13.688	89	3.05	0.411	1.75	2.37	5.64	68
FWPHBR04	FWPHBR-G025	G025	5.8	12.7478	432	2.02	0.0168	0.311	2.07	2.53	380
FWPHBR04	FWPHBR-G026	G026	0.041	4.024	58	0.262	0.031	1.07	2.47	0.15	39
FWPHBR04	FWPHBR-G027	G027	0.439	3.31385	111	0.379	0.01785	0.805	1.02	0.653	97
FWPHBR04	FWPHBR-G022	G022	0.0202	2.9712	45	0.272	0.217	1.79	1.24	0.222	24

Different cell colors behind the analyte names on the [AnalyteSummary] worksheet represent different categories of analytes

The [DataTable] worksheet (shown below) holds the final data set omitting all screened data and analytes and including the new groups. This is the data set that will be imported into the FPMAnova.xls spreadsheet.

Data Preparation for Floating Percentile Calculations										
Page 5: Data Set After Screening and Summing										
Survey	Station	Sample	2,4'-DDT	2-Methylnaphthalene	4-Methylphenol	Acenaphthene	Acenaphthylene	Acetone	Aldrin	al
			PPB	PPB	PPB	PPB	PPB	PPB	PPB	
FWPHBR04	FWPHBR-D1-1	D1-1	0.336	1.7	6.6	1.4	3.4			
FWPHBR04	FWPHBR-D2	D2	0.194	2.6		4.6	13			
FWPHBR04	FWPHBR-G007-1	G007-1	0.037	3.2		5.7	3.5			
FWPHBR04	FWPHBR-G009	G009	3.38	5.5		8.1	9.7			
FWPHBR04	FWPHBR-G010	G010	0.854	4.9		4.1	6.7			
FWPHBR04	FWPHBR-G011	G011	1.99	7.5	5.5	8.8	16			
FWPHBR04	FWPHBR-G015	G015	2.84	9.6		9.7	36			
FWPHBR04	FWPHBR-G017	G017	1.83	8.5		8.2	19			
FWPHBR04	FWPHBR-G019	G019	1.96	6.8		4.9	12		0.902	
FWPHBR04	FWPHBR-G020	G020	1.13	10	9.7	18	10			
FWPHBR04	FWPHBR-G024	G024	0.467	54		98	12		1.38	

The [DataTable] worksheet in the FPMData.xls spreadsheet

The [DataSetSummary] worksheet (shown below), contains a table that summarizes the number of analytes and data points that were screened out or retained for various reasons. It also contains a list of the name and members of every new group that you defined as well as a list of every analyte that was screened out.

As described previously, instructions are provided on this page for saving or retrieving group information. A special file, FPMDataGroups.xls, is used for this purpose. FPMDataGroups.xls can hold up to twenty sets of group information. If you need room for additional groups you can make a copy of this file with a different name. It must be a copy so that it contains the macros that are in the original file. FPMDataGroups.xls is described in the next section.

Data Preparation for Floating Percentile Calculations

Page 6: Data Set Summary

Description of Data Set		Members of Defined Groups	Screened Analytes	Saving and Reusing Defined Groups
Number of Samples	308			
Summary by Analytes				
No. of Analytes in Initial Data Set	147	Group One	2,4'-DDD	<p>1. If you want to retain the groups defined in the tables on the [CreateSums] page so that you can use them with the next set of analytes, put a check mark in the Retain Group List box on that page. This will do two things:</p> <p>a. It will prevent the Members of Defined Groups list on this page from being deleted when the Clear Previous button is used on the [InputData] page, and</p> <p>b. It will recreate the group names and numbers on the [CreateSums] page when the CreateAnalyteList button on that page is used.</p> <p>2. If you want to save the current groups for future use on this or other FPMData spreadsheets, do the following:</p> <p>a. Enter a name for the group in the Save Current Group table below.</p> <p>b. Enter the name of the file into which the group data will be stored in the Save Current Group table below.</p> <p>c. Make sure that the storage file is open in the same folder as this spreadsheet, then click on the Save Group button.</p>
No. of New Analyte Groups	2	2,4'-DDE	2,4'-DDT	
Total Analytes and Groups	149	2,4'-DDT	4,4'-DDD	
Analytes Appearing in the Final Data Table		4,4'-DDD	4,4'-DDE	
Unsummed Analytes	84	4,4'-DDE	4,4'-DDT	
Summed w/ sufficient data	1	4,4'-DDT	Aroclor 1248	
Groups w/ sufficient data	2	4,4'-DDE	Aroclor 1254	
Number in Final Data Table	87	4,4'-DDT	Aroclor 1260	
Analytes Omitted from Final Data Table		Group Two	1,2,3-Trichloropropane	
Initial Analytes w/ too few data	50	Aroclor 1016	1,2,4-Trichlorobenzene	
Groups w/ too few data	0	Aroclor 1221	1,2-Dichlorobenzene	<p>Save Current Group</p> <p>Enter a name for the group below.</p> <p>Another Group</p> <p>Enter the name of the storage file below.</p> <p>FPMDataGroups.xls</p> <p>Save Group</p>
Summed & not retained (ok data)	12	Aroclor 1242	1,2-Dichloroethane	
Omitted by choice (ok data)	0	Aroclor 1248	1,4-Dichlorobenzene	
Number Omitted from Final Data Table	62	Aroclor 1254	1-Methylnaphthalene	
Total Analytes and Groups	149	Aroclor 1260	2,3,4,5-Tetrachlorophenol	
Summary by Data Points		Aroclor 1268	2,4,5-Trichlorophenol	
Initial Number of Data	16691		2,4-D	
No. of Data in New Groups	461		2,4-DB	
Total No. of Available Data	17152		2,4-Dichlorophenol	
Data Appearing in the Final Data Table			2,4-Dimethylphenol	
Unsummed Data	14395		2,4-Dinitrotoluene	
Data summed & retained (ok data)	238		2-Chlorophenol	
Data in new groups retained	461		2-Methylphenol	
Number in Final Data Table	15094		4-Chloro-3-methylphenol	
Data Omitted from Final Data Table			4-Nitroaniline	
Initial Data (too few data)	389		alpha-Chlordane	
Summed & not retained	1669		Aniline	
Omitted by choice	0		Benzene	
Data w/ bad DGs	0		Benzoic acid	
Data in new groups omitted	0		Benzyl alcohol	
Number Omitted from Final Data Table	2058		Bis(2-chloroethyl) ether	
Total No. of Available Data	17152		Caprolactam	
Program Run Time (sec)			Carbon disulfide	
	92		Chlorobenzene	
			Chloroform	
			Diethyl phthalate	
			Dimethyl phthalate	
			Di-n-octyl phthalate	

The [DataSetSummary] worksheet in the FPMData.xls spreadsheet

2.2 FPMDataGroups.xls

2.2.1 Overview

FPMDataGroups is an optional spreadsheet that can be used for group information defined in FPMData.xls. This spreadsheet:

1. Accepts and stores up to twenty sets of group name and analyte list information created in FPMData.xls; and
2. Restores previously saved group name and analyte list information to FPMData.xls.

2.2.2 Worksheets

This spreadsheet contains only one worksheet:

“**FPMDataGroups**” contains twenty tables for holding group name and analyte list information. It also contains one table that lists the names given to the groups when they were saved.

2.2.3 Instructions

The worksheet in FPMDataGroups.xls is shown below. Detailed instructions for using this spreadsheet are given on the [DataSetSummary] page of FPMData.xls. They are also listed in Section 2.1.4 of this guidance document as part of the instructions for FPMData.xls.

Data Preparation for Floating Percentile Calculations			
Data Group Storage			
Groups Currently Stored in this Worksheet		Group 1	Group 2
		Complete DDT and Aroclor	Partial DDT and Aroclor
1	Complete DDT and Aroclor	DDTs	DDTs
2	Partial DDT and Aroclor	2,4'-DDD	2,4'-DDD
3		2,4'-DDE	2,4'-DDE
4		2,4'-DDT	2,4'-DDT
5		4,4'-DDD	
6		4,4'-DDE	Aroclors
7		4,4'-DDT	Aroclor 1016
8		Aroclors	Aroclor 1221
9		Aroclor 1016	Aroclor 1242
10		Aroclor 1221	
11		Aroclor 1242	
12		Aroclor 1248	
13		Aroclor 1254	
14		Aroclor 1260	
15		Aroclor 1268	
16			
17			
18			
19			
20			
To delete group information, delete the name of the group from the list above and then click on the Refresh Worksheet button.			
Refresh Worksheet			

The main thing to remember when using this spreadsheet is that both FPMData.xls and FPMDataGroups.xls must be open in the same folder on your computer in order for the save and retrieve procedures to work. The files do not need to have these exact names as long as they are copies of the original files. Therefore, if you need to save information for more than 20 groups you can store them in copies of FPMDataGroups.xls that have different names.

If you want to delete some previously saved group information, delete the name of that group from the table of "Groups Currently Stored in this Worksheet" and then click on the "Refresh Worksheet" button.

2.3 *FPMAnova.xls*

2.3.1 Overview

FPMAnova is the second of three spreadsheets used for implementing the FPM. It compares the hit and no-hit distributions for each analyte by calculating an analysis of variance and listing results for several different levels of significance. This spreadsheet:

1. Accepts screened analytical data from FPMData.xls;
2. Requires one or more sets of bioassay hit/no-hit results with the same survey, station, and sample IDs as the analytical data;
3. Sorts the data by analyte into hit and no-hit distributions;
4. Stores the sorted hit/no-hit data in another worksheet, if desired;
5. Calculates analysis of variance on the hit versus no-hit distributions to determine whether each analyte is associated with toxicity in the data set; and
6. Allows the user to select analytes based on the ANOVA results and creates a data set for export into FPMCalc.xls.

Before running this program you must have the AnalysisToolPack and AnalysisToolPack-VBA Add-Ins installed in your copy of Excel. From the "Tools" menu, select "Add-Ins," then put a check mark next to AnalysisToolPack and AnalysisToolPack-VBA and click OK. These Add-Ins have different file names in Office 2003 and 2007; therefore, you must have the correct FPMAnova.xls version for the version of Office that you are using.

2.3.2 Worksheets

This spreadsheet workbook consists of 27 worksheets. The following 6 are used on a regular basis:

"ControlScreen" contains most of the instructions for using the spreadsheet along with buttons for running the program, loading FPMData, and clearing data from the worksheets.

"ChemData" holds the data from the FPMData spreadsheet in preparation for calculating the analysis of variance of the hit and no-hit data.

"BioData" holds up to 20 sets of biological hit/no-hit designations that correspond to the analytical data results on the [ChemData] page. These hit/no-hit designations may be for different tests, endpoints, or levels of effects.

"SortedData" temporarily stores the analytical data sorted into hit and no-hit columns that will be used for the analysis of variance.

"AnovaResults" stores the results of the analysis of variance calculations for all of the sets of bio data stored on the [BioData] page.

"FinalDataSet" stores the data that have been selected for use in the FPMCalc spreadsheet.

There are an additional 20 worksheets that are initially hidden but are available for storing data: These worksheets are named **1, 2, 3, ... etc. up to 20**, and are used only if, when you are running the program, you decide to save all of the lists of data that have been sorted into hit and no-hit groups during the ANOVA

calculations. This can be done by putting a “Y” in the appropriate cell on the [ControlScreen] page. If the sorted data sets are saved, a worksheet is unhidden for each data set and is renamed with the name of the data set.

The following worksheet is not used for the calculations and is normally hidden at all times:

“Names” is a reference worksheet that lists all of the named cells used in all of the worksheets. It also contains a table used to store the names of the tool bars that are open when the program begins.

This workbook includes 4 modules that contain the main macro code for running the entire workbook. The forms and the macro modules are password-protected; however, none of the worksheets in the workbook are protected so you must be careful not to paste data into cells that may already contain formulas. If you follow the instructions on each worksheet page, that should not be a problem.

2.3.3 Instructions

Instructions for running this program along with descriptions of and images from the FPMAnova spreadsheet are given below. Instructions are also provided on the [ControlScreen] and [AnovaResults] worksheets in the body of the spreadsheet.

As noted in the section on Spreadsheet Security, when you open the spreadsheet you will see a warning that the spreadsheet contains macros. Be sure to click on the “Enable Macros” button if you are running the ANOVA macros. However, if you are just opening the spreadsheet to copy the results into the FPMCals.xls spreadsheet, it will open and close more quickly if you select “Disable Macros.”

The [ControlScreen] worksheet (shown below) primarily consists of instructions for running the program and buttons for clearing data from various worksheets within the spreadsheet. If you plan to work with a new data set that is not yet in the spreadsheet you should use the “Clear All Workbook Tables” button to make sure that no data from a previous data set remain. If you are using data that are already in the spreadsheet, use some of the other “Clear ...” buttons to delete any old results that you would like to replace.

Analysis of Variance Calculations

Page 1: Control Screen

FPMAnova.xls Version 072908

Before running this program the you must have the [AnalysisToolPack](#) and [AnalysisToolPack-VBA](#) Add-Ins installed in your copy of Excel. From the "Tools" menu, select "Add-Ins," then put a check mark next to AnalysisToolPack and AnalysisToolPack-VBA and click OK.

Instructions	Anova Results	Run ANOVA
<p>1. Use the FPMData.xls spreadsheet to screen and group the raw analytical data.</p> <p>2. Use the Copy Screened Data button on this page to copy data from the FPMData [DataTable] page and paste it into the [ChemData] page in this spreadsheet. (See instructions at the right.)</p> <p>[Note: If necessary, you can copy and paste the data manually. However, if you do this be sure that all old data have been deleted and that all new data are in the correct format and the correct columns.]</p> <p>3. Insert the results of the biological tests into the [BioData] page. You can insert up to 20 different sets of results but the survey, station, and sample labels MUST BE THE SAME as those found on the [ChemData] page.</p> <p>4. Click on the Run ANOVA button. Results of the tests will be on the [AnovaResults] page. The meanings of the results are explained in the adjacent text box.</p> <p>Buttons are provided to clear some or all of the data tables when needed.</p>	<p>This Analysis of Variance test compares the Hit and No-Hit data distributions for each analyte and returns one of the following results:</p> <p>0 = This analyte shows no significant difference in its hit and no-hit distributions.</p> <p>0* = This analyte shows significant differences in its hit and no-hit distributions at $p < 0.1$.</p> <p>1 = This analyte shows significant differences in its hit and no-hit distributions at $p < 0.05$.</p> <p>1* = This analyte shows significant differences in its hit and no-hit distributions at $p < 0.005$.</p> <p>1** = This analyte shows significant differences in its hit and no-hit distributions at $p < 0.0005$.</p> <p>[Note: The final sorted set of data is the only one stored on the [SortedData] page. If you want to save sorted data from all sets of bio data, put "Y" into the Save Sorted Data box at the right and the data will be stored on extra pages. If not, put "N."</p> <p>Run Time (sec) 16.0</p>	<p>Run ANOVA</p> <p>Clear All Workbook Tables</p> <p>Clear ChemData Table</p> <p>Clear BioData Table</p> <p>Clear SortedData Table</p> <p>Clear AnovaResults Table</p> <p>Save Sorted Data? (Y or N) Y</p> <p>Clear Saved Sorted Data Tables</p> <p>To use Copy Screened Data:</p> <p>1. The file containing the screened data must be the FPMData.xls file or an exact copy (i.e., worksheets and tables in that file must have the formats and names defined in FPMData.xls.)</p> <p>2. Open the screened data file in the same folder as this ANOVA file.</p> <p>3. Enter the name of the data file below.</p> <p>4. Click on the Copy Screened Data button.</p> <p>FPMData-CH10G.xls</p> <p>Copy Screened Data</p>

The [ControlScreen] worksheet in the FPMAnova.xls spreadsheet

The [ChemData] worksheet on the FPMAnova spreadsheet is set up to accept the data from the [DataTable] worksheet of the FPMData spreadsheet. To do this you must have both of the spreadsheets stored in the same folder on your computer and both spreadsheets must be open. Then type the complete name of your copy of the FPMData spreadsheet (for example, FPMData-CH10G.xls) into the white cell near the bottom right corner of the [ControlScreen] page and click on the "Copy Screened Data" button right below that cell. When the transfer is complete the data will be located on the [ChemData] page (shown below). You can then close the FPMData spreadsheet.

Analysis of Variance Calculations

Page 2: Analytical Data

Survey	Station	Sample	4-Methylphenol	Acetone	Aldrin	alpha-Hexachlorocyclohexane	Ammonia	Antimony	Arsenic
			PPB	PPB	PPB	PPB	PPM	PPM	PPM
FWDMMPO5	FWDMMPO-WRPG106	WRPG106	58				137	0.88	4.34
FWDMMPO5	FWDMMPO-WRPG13	WRPG13					153	0.08	4.14
FWDMMPO5	FWDMMPO-WRPG16	WRPG16					143	0.18	3.85
FWDMMPO5	FWDMMPO-WRPG48	WRPG48					102	0.66	6.19
FWDMMPO5	FWDMMPO-WRPG54	WRPG54					201	0.11	4.44
FWDMMPO5	FWDMMPO-WRPG66	WRPG66					78.1	0.14	4.21
FWDMMPO5	FWDMMPO-WRPGREF03	WRPGREF03	13				95.3	0.07	2.84
FWDMMPO5	FWDMMPO-WRVC05	WRVC05	28				184	0.12	3.42
FWDMMPO5	FWDMMPO-WRVC07	WRVC07	19				297	0.11	3.21
FWDMMPO5	FWDMMPO-WRVC106	WRVC106	28				775	0.62	4.69
FWDMMPO5	FWDMMPO-WRVC11	WRVC11	20				328	0.12	2.77
FWDMMPO5	FWDMMPO-WRVC12	WRVC12	70				417	0.12	4.09

The [ChemData] worksheet in the FPMAnova.xls spreadsheet

Enter up to 20 corresponding sets of hit/no-hit data into the table on the [BioData] worksheet (shown below). The sets of Survey, Station, and Sample values for the bio data must be identical to the Survey, Station, and Sample values of the analytical data so the program can match corresponding sets of data on which to perform an analysis of variance. You do not have to paste the bio data into the [BioData] table in the same row order as they are shown in the [ChemData] worksheet. The program will sort the data sets and put them into the correct order.

Analysis of Variance Calculations										
Page 3: Biological Data										
Number of Hits (1)	62	46								
Number of NoHits (0)	442	458								
Number of Interdeterminates (-1)	0	0								
Total Number of Reported Results	504	504								
Survey	Station	Sample	SQS-CH10G	CSL-CH10G						
FWDMMPO5	FWDMMPO-WRPG106	WRPG106	0	0						
FWDMMPO5	FWDMMPO-WRPG13	WRPG13	0	0						
FWDMMPO5	FWDMMPO-WRPG16	WRPG16	0	0						
FWDMMPO5	FWDMMPO-WRPG48	WRPG48	0	0						
FWDMMPO5	FWDMMPO-WRPG54	WRPG54	0	0						
FWDMMPO5	FWDMMPO-WRPG66	WRPG66	0	0						
FWDMMPO5	FWDMMPO-WRPGREF03	WRPGREF03	0	0						
FWDMMPO5	FWDMMPO-WRVC05	WRVC05	0	0						

The [BioData] worksheet in the FPMAnova.xls spreadsheet

Go back to the [ControlScreen] page and click on the "Run Anova" button. When the program starts it will check for certain errors or omissions. If one is encountered the program will terminate and a message box will appear on the screen telling you what is wrong. As the program runs, it will sort the analytical data into separate hit and no-hit columns and place it on the [SortedData] worksheet (shown below) prior to performing the analysis of variance. If you have more than one set of bio data, the sorted data for the current set will replace the sorted data for the previous set on the [SortedData] page. If you want to save all of the individual sets of sorted data, put a "Y" on the "Save Sorted Data?" line on the [ControlScreen] page.

Analysis of Variance Calculations											
Page 4: Analytical Data Sorted By Hit / No-Hit Results											
The data on this page are for the most recently run set of bio data only. To save data from all runs, answer "Y" to "Save Sorted Data?" on [ControlScreen].											
4-Methylphenol		Acetone		Aldrin		alpha-Hexachlorocyclohexane		Ammonia		Antimony	
Hit	NoHit	Hit	NoHit	Hit	NoHit	Hit	NoHit	Hit	NoHit	Hit	NoHit
48	4	2.5	2.6	0.173	0.052	0.093	0.047	0.77	0.05	0.1	0.05
52	4.9	4.2	3.2	2.19	0.091	0.995	0.063	2.91	0.24	0.1	0.06
73	5.5	7.5	3.6	6	0.116	1.04	0.068	2.91	0.35	0.13	0.06
80	5.7	11	3.6	7.08	0.143	10	0.069	6.77	0.47	0.13	0.06
83	5.8	11	4	132	0.155		0.069	22.9	0.48	0.14	0.07
150	6	11	4		0.156		0.07	32	0.56	0.19	0.07
215	6	12	4		0.166		0.091	49.9	0.59	0.2	0.07
382	6.4	14	4		0.204		0.091	52.8	0.62	0.21	0.07
710	6.6	14	4.1		0.207		0.095	67	0.73	0.23	0.07
1560	6.6	27	4.1		0.214		0.095	74.7	0.81	0.23	0.07

The [SortedData] worksheet in the FPMAnova.xls spreadsheet

If you save the sets of sorted data, new worksheets will appear during the course of the calculations. They can be identified by the data set name shown on the tabs of the new worksheets (see example below).

Analysis of Variance Calculations

Page 6: Final Floating Percentile Data Set

Hit/NoHit	Survey	Station	Sample	4-Methylphenol	Acetone	Aldrin	alpha-Hexachlorocyclohexane	Ammonia	Al
CSL-CH10G				PPB	PPB	PPB	PPB	PPM	
0	FWDMMMP05	WDMMP-WRPG1	WRPG106	58				137	
0	FWDMMMP05	WDMMP-WRPG1	WRPG13					153	
0	FWDMMMP05	WDMMP-WRPG1	WRPG16					143	
0	FWDMMMP05	WDMMP-WRPG4	WRPG48					102	
0	FWDMMMP05	WDMMP-WRPG6	WRPG64					201	
0	FWDMMMP05	WDMMP-WRPG6	WRPG66					78.1	
0	FWDMMMP05	DMMPWRPGREF	WRPGREF03	13				95.3	
0	FWDMMMP05	WDMMP-WRVC0	WRVC05	28				184	
0	FWDMMMP05	WDMMP-WRVC0	WRVC07	19				297	
0	FWDMMMP05	WDMMP-WRVC1	WRVC106	28				775	
0	FWDMMMP05	WDMMP-WRVC1	WRVC11	20				328	
0	FWDMMMP05	WDMMP-WRVC1	WRVC12	70				417	

The [AnovaResults] worksheet in the FPMAnova.xls spreadsheet

As noted earlier, if you opted to save the sets of sorted data, extra worksheets will be opened during the run and each set of sorted data will be placed into a separate worksheet with the name of the data set on the worksheet tab.

2.4 *FPMCalc.xls*

2.4.1 Overview

FPMCalc is the third of three spreadsheets used for implementing the FPM. This spreadsheet carries out the actual FPM calculations using the final data set that was screened with FPMData and tested for analysis of variance with FPMAnova. FPMCalc.xls:

1. Accepts the final table of chem and bio data created in FPMAnova.xls;
2. Sorts the data by analyte into distributions from lowest to highest concentrations;
3. Uses the data distributions to generate data distribution percentiles;
4. Selects an initial FPM dataset based on user-entered targets of %False Negatives;
5. Compares the selected FPM dataset to the chem and bio data and counts the number of false positives and false negatives;
6. Modifies the dataset in order to maintain the target %False Negatives while minimizing the %False Positives;
7. Computes a variety of reliability indices to evaluate the reliability of the results; and
8. Allows comparison of alternative SQG sets and their reliabilities to the results of the FPM model.

2.4.2 Worksheets

This spreadsheet workbook consists of 8 worksheets:

“**ControlScreen**” contains the instructions for using the spreadsheet along with buttons for running the program, loading FPMAnova results, and clearing data from the worksheets.

“**DataTable**” holds the biological and analytical data that have been analyzed and assembled in FPMAnova.

“**Distributions**” lists the analytical data by analyte and displays it in order from lowest to highest concentration.

“**Percentiles**” lists the percentile concentrations for each analyte based on the previous distributions.

“**ErrorCalc**” lists the number of True/False Hits/NoHits that would result for each of the previous percentile concentrations if those concentrations were used as the sediment standards.

“**Criteria**” contains the initial and final FPM results for the range of %False Negative Targets specified in the table on the [ControlScreen] page.

“**DataStorage**” is not used in any of the calculations. It is provided for the user to store and compare results from different sets of FPM calculations

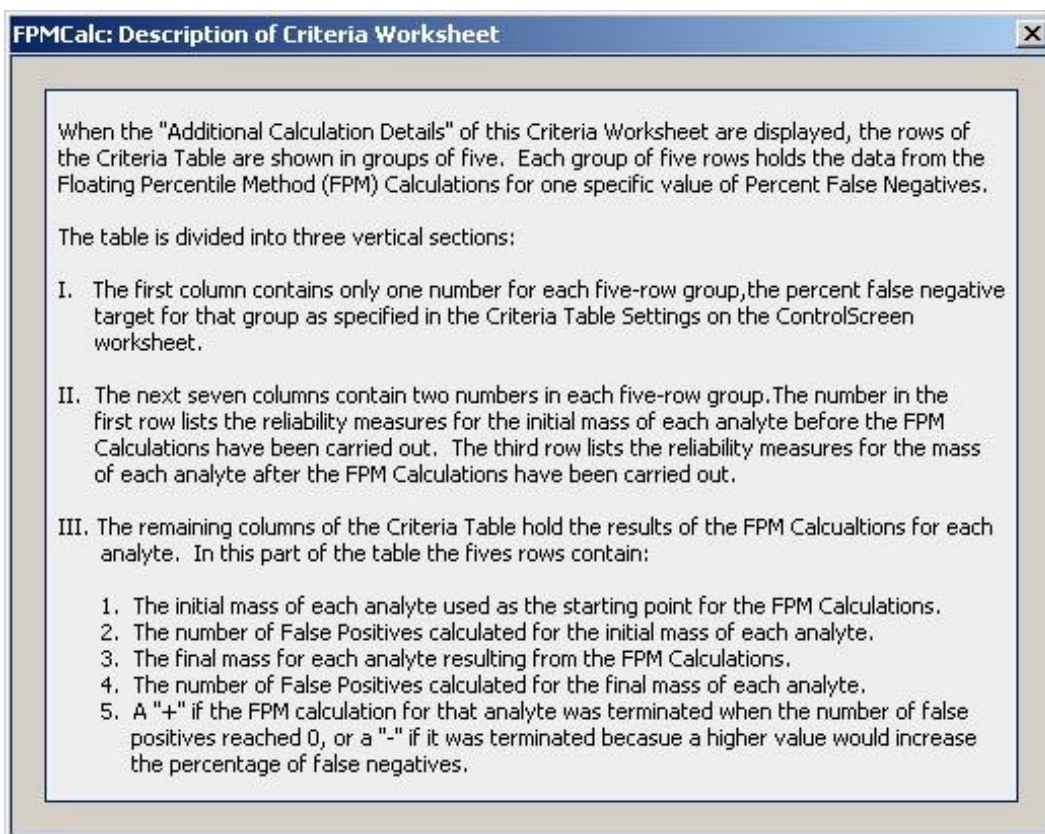
“**Names**” is a reference worksheet that lists all of the named cells used in all of the worksheets. It is a hidden page that is never shown to the regular user.

This workbook also contains 2 forms (**Description of Criteria Worksheet** and **Reliability Measures Definitions**). The forms provide additional information about the [Criteria] page and can be opened by clicking on the buttons located on that page.

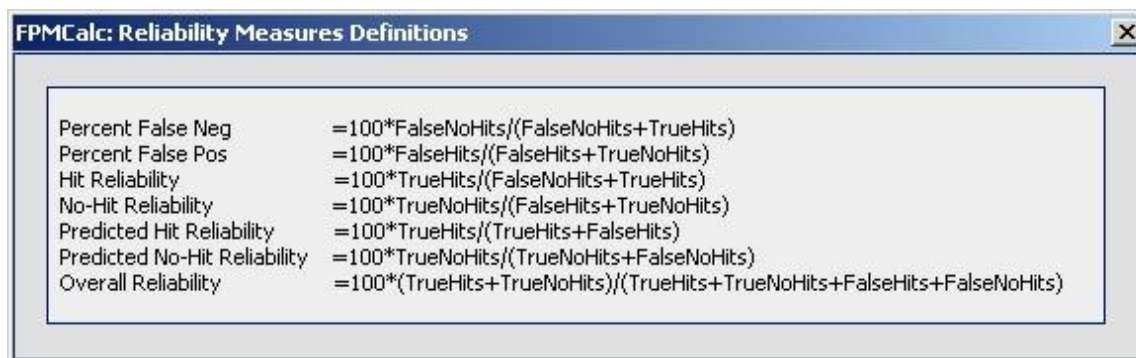
Finally, this workbook contains 10 modules that contain the main macro code for running the entire workbook.

2.4.3 Forms

2.4.3.1 Description of Criteria Worksheet



2.4.3.2 Reliability Measures Definitions



2.4.4 Instructions

Instructions for running this program along with descriptions of and images from the FPMCalc spreadsheet are given below. Instructions are also provided on the [ControlScreen] worksheet (shown below) in the body of the spreadsheet. As noted in the section on Spreadsheet Security, when you open the spreadsheet you will see a warning that the spreadsheet contains macros. Be sure to click on the “Enable Macros” button if you are running the model. However, if you just want to look at existing results, it is faster to disable the macros.

Floating Percentile Calculations																																												
Page 1: Control Screen																																												
FPMCalc.xls Version 120608																																												
<p>Instructions</p> <ol style="list-style-type: none"> 1. Use FPMAnova.xls to determine what set of biological (Hit/No-Hit) data and analytes to use for the Floating Percentile Method (FPM) calculations. 2. Use the Copy Anova Data button on this page to copy data from the FPMAnova [FinalDataSet] page and paste it into the [DataTable] page in this spreadsheet. (See <i>instructions at the right</i>.) <p>[Note: If necessary, you can copy and paste the data manually. However, if you do this be sure that all old data have been deleted and that all new data are in the correct format.]</p> <ol style="list-style-type: none"> 3. You can carry out FPM Calculations for up to 10 different false negative target values at one time. Enter initial and final false negative target values under Criteria Table Settings along with the interval to use in the calculations. The value of $((Final - Initial) / Interval)$ cannot exceed 10. 4. Indicate in the AET and Outlier Settings table if you want to Pre-Screen for AETs. If you do, the program determines the AET of each analyte before starting the calculations. If the number of false positives = 0 for an AET, the result for that analyte is set at the AET and it is omitted from the subsequent FPM calculations. 	<ol style="list-style-type: none"> 5. If you want to omit outliers from the AET evaluations, indicate that in the AET and Outlier Settings table. You also must specify how much larger an outlier must be compared to the next closest data point by listing a multiplier in the AET and Outlier Settings table. Outliers are then excluded from the AET evaluations. 6. Use the Calculate Floating Percentiles button to start the calculations. <p>To use the Copy Anova Data button:</p> <ol style="list-style-type: none"> 1. The file containing the data must be the FPMAnova.xls file or an exact copy (<i>worksheets and tables in that file must have the same formats and names defined in FPMAnova</i>). 2. Open the Anova file in the same folder as this Floating Percentile file. 3. Enter the name of the Anova file in the cell under "Source File" below. 4. Click on the Copy Anova Data button. 	<p>Data Summary and Program Settings</p> <table border="1"> <thead> <tr> <th colspan="2">Chemical Data Summary</th> </tr> </thead> <tbody> <tr> <td>Number of Analytes</td> <td>32</td> </tr> <tr> <td>Analytes Screened as AET</td> <td>0</td> </tr> <tr> <td>Analytes Calculated by FPM</td> <td>32</td> </tr> <tr> <td>No. of Data Points</td> <td>3035</td> </tr> <tr> <td>Number of Samples</td> <td>185</td> </tr> <tr> <td>Number of Hits</td> <td>61</td> </tr> <tr> <td>Number of Indeterminates</td> <td>0</td> </tr> <tr> <td>Number of No-Hits</td> <td>124</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="2">Criteria Table Settings</th> </tr> </thead> <tbody> <tr> <td>Initial False Negative Target</td> <td>0</td> </tr> <tr> <td>Final False Negative Target</td> <td>30</td> </tr> <tr> <td>Target Interval</td> <td>5</td> </tr> <tr> <td>Number of Steps</td> <td>6</td> </tr> <tr> <td>Number of Increments</td> <td>10</td> </tr> <tr> <td>Percent Precision</td> <td>10</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="2">AET and Outlier Settings</th> </tr> </thead> <tbody> <tr> <td>Pre-Screen AETs (Y or N)?</td> <td>N</td> </tr> <tr> <td>Omit AET Outliers (Y or N)?</td> <td>N</td> </tr> <tr> <td>Outlier exceeds by multiple</td> <td>3</td> </tr> <tr> <td>Program Run Time (minutes)</td> <td>12.1</td> </tr> </tbody> </table> <p>Clear All Worksheets</p> <p>Clear All Sheets Except Data & Storage Tables</p> <p>Watch the Calculations (Y or N) ? N</p> <p>Calculate Floating Percentiles</p>	Chemical Data Summary		Number of Analytes	32	Analytes Screened as AET	0	Analytes Calculated by FPM	32	No. of Data Points	3035	Number of Samples	185	Number of Hits	61	Number of Indeterminates	0	Number of No-Hits	124	Criteria Table Settings		Initial False Negative Target	0	Final False Negative Target	30	Target Interval	5	Number of Steps	6	Number of Increments	10	Percent Precision	10	AET and Outlier Settings		Pre-Screen AETs (Y or N)?	N	Omit AET Outliers (Y or N)?	N	Outlier exceeds by multiple	3	Program Run Time (minutes)	12.1
Chemical Data Summary																																												
Number of Analytes	32																																											
Analytes Screened as AET	0																																											
Analytes Calculated by FPM	32																																											
No. of Data Points	3035																																											
Number of Samples	185																																											
Number of Hits	61																																											
Number of Indeterminates	0																																											
Number of No-Hits	124																																											
Criteria Table Settings																																												
Initial False Negative Target	0																																											
Final False Negative Target	30																																											
Target Interval	5																																											
Number of Steps	6																																											
Number of Increments	10																																											
Percent Precision	10																																											
AET and Outlier Settings																																												
Pre-Screen AETs (Y or N)?	N																																											
Omit AET Outliers (Y or N)?	N																																											
Outlier exceeds by multiple	3																																											
Program Run Time (minutes)	12.1																																											
<p>Source File</p> <p>FPMAnova-HY28M-SL2.xls</p> <p>Copy Anova Data</p>																																												

The [ControlScreen] worksheet in the FPMCalc.xls spreadsheet

[ControlScreen] contains several buttons such as “Copy Anova Data” and “Clear All Worksheets” that perform specific tasks to prepare the spreadsheet for analysis as well as the “Calculate Floating Percentiles” button that initiates the main analytical function of the program. There is also a table where you can change certain variables that affect how the program operates and a table that summarizes the number and types of data in the most recently analyzed data set. These will be discussed later.

If you plan to work with a data set that is already in the spreadsheet you should click on the “Clear All Sheets Except Data & Storage Tables” button to delete the results of the previous analysis. If you want to work with a new data set you will, of course, have to load the relevant data. To make sure that no data from an earlier data set remain in any of the tables, you should first click on the “Clear All Worksheets” button.

The [DataTable] page on this spreadsheet (see next page) is set up to accept the data from the [FinalDataSet] page of the FPMAnova spreadsheet. For this to work properly, you must have both of the spreadsheets stored in the same folder on your computer, and both spreadsheets must be open. Then type the complete name of your copy of the FPMAnova spreadsheet (for example, FPMAnova-HY28M-SL2.xls) into the white cell near the bottom

in the center of the [ControlScreen] page and then click on the “Copy Anova Data” button right below that cell. When the transfer is complete the data will be located on the [DataTable] page. You can then close the FPMAnova spreadsheet.

Floating Percentile Calculations								
Page 2: Table of Biological and Analytical Data								
Hit/NoHit	Survey	Station	Sample	4-Methylphenol	Acetone	alpha-Hexachlorocyclohexane	Ammonia	Antimony
CSL-CH10M				PPB	PPB	PPB	PPM	PPM
0	LUUCS000	527	L17656-18				37.3	1.6
0	LUUCS000	535	L17656-6				23.9	0.66
0	LKWA00	544	L18812-1				7.19	
0	LUUCS000	544	L17656-3				29.4	
0	LKWA00	560	L18812-2				25.1	
0	LKWA00	804	L18493-2				138	
0	LKWA00	826	L18493-3				124	
0	LKWA00	831	L18493-9				94.9	
0	LKWA00	834	L18812-3				54.6	
0	LKWA00	840	L18493-10				101	
0	LKWA00	852	L18493-4				149	
0	LKWA00	861	L18493-5				143	
0	LKWA00	862	L18656-6				3.9	
0	LKWA00	862	L18656-7				3.6	
0	LKWA00	890	L18493-7				135	
0	LKWA00	0425A	L18656-2				56.5	
0	LKWA00	0611A	L18493-11				103	
0	LKWA00	0801A	L18493-1				23.6	
0	LKWA00	0807A	L18656-5				31.9	
0	LKWA00	0814B	L18656-10				71.4	
0	LKWA00	0817A	L18656-3				59.3	
0	LKWA00	0829A	L18812-6				45.7	
0	LKWA00	0864A	L18862-2				60.2	0.4
0	SALIII97	10B2	97218311	16				
0	SALIII97	2C2	97218283	699				
0	SALIII97	3C3	97218287	239				
0	LKWA00	4901A	L18656-4				22.8	
0	SALIII97	4B2	97218288	476				
0	SALIII97	4C2	97218290	159				

The [DataTable] worksheet in the FPMCalc.xls spreadsheet

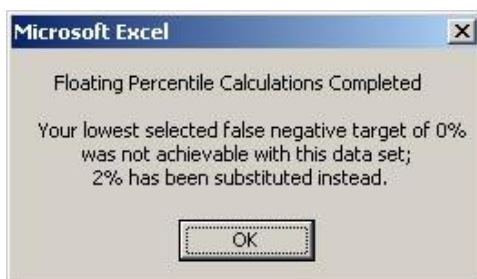
When sediment data are compared to sediment quality guidelines (SQGs), some of the data will exceed the SQGs (“hits” or “positives”) and some will be less than or equal to the SQGs (“no-hits” or “negatives”). By comparing proposed SQGs to results from biological tests in which hits and no-hits are based on factors such as growth or mortality rates, we can determine if the SQGs would generate “true hits” (true positives; the SQG predicts a hit and the biological test results also show a hit), “true no-hits” (true negatives; the SQG predicts a no-hit and the biological test results also show a no-hit), “false hits” (false positives; the SQG predicts a hit but the biological test results show a no-hit), or “false no-hits” (false negatives; the SQG predicts a no-hit but the biological test results show a hit). These four potential outcomes are illustrated in the table below. The goal of the Floating Percentile Method is to generate a set of SQGs that minimizes both the false negatives and the false positives and gives the greatest number of correct predictions.

		Sediment Standards	
		Hit	No-Hit
Results of Biological Tests	Hit	True Hit or True Positive	False No-Hit or False Negative
	No-Hit	False Hit or False Positive	True No-Hit or True Negative

Before running the FPMCalc spreadsheet you must specify the percentage of false no-hits (also known as percent false negatives, %FN) that you would like to obtain. You can do this for up to 10 different %FN values in a single run of the spreadsheet to compare the results at different levels. A range of 5-25% is typical.

Put the lowest %FN value in the Initial False Negative Target, the highest value in the Final False Negative Target, and how much to increase the value of each step in the Target Interval. For example, if you want results for 5%, 10%, and 15% FN you would use 5 for the initial value, 15 for the final value, and 5 for the interval.

If you select an FN target that is lower than the lowest %FN for the given data set, the program will substitute the first integral value above the lowest available %FN (this typically only happens for 0%FN, and rarely). Calculations will continue without interruption but a message will be posted at the end of the run to inform you that a substitution was made.



During the FPM calculations the program restricts the tentative standards to the range of known concentrations for each analyte. Instead of testing every available concentration, the calculations are done iteratively and the program attempts to converge on the best result for each analyte. You control the size of the steps taken during the iterations by specifying the number of increments to use for the range of concentrations. The larger the number of increments, the smaller the size of each increment, and the longer the program will take to obtain a result. If the number of increments is too small, however, the size of the increment is large and the program will keep jumping past its ultimate target and will take longer to converge on the best result.

Number of Increments	Time of Analysis (min)
2	46.3
3	33.4
4	30.7
5	29.7
6	29.7
7	29.8
8	31.3
9	31.8
10	32.6

The table above shows results of program runtimes versus the number of increments for a selected data set. These results will vary from one data set to another. You may want to experiment with this value but for consistency with the RSET FPM calculations we recommend that you leave the "Number of Increments" set to 10.

The initial concentrations used in the FPM calculations are based on the distribution of the available analytical data and the target %FN. The program sorts the data by analyte and creates a table with the data distributed from lowest to highest for each analyte (see [Distributions] worksheet below). It then calculates percentile distributions for each analyte (see [Percentiles] worksheet below) and calculates %FN and other parameters for each percentile row (see [ErrorCalc] worksheet below). The program finds the highest percentile on the [ErrorCalc] table that does not exceed the target %FN and takes the concentrations from the corresponding percentile row in the [Percentiles] table, using that as the starting point for the model calculations.

Floating Percentile Calculations									
Page 3: Distribution of Analytical Data from Lowest to Highest									
4-Methylphenol	Acetone	alpha-Hexachlorocyclohexane	Ammonia	Antimony	Arsenic	Benzoic acid	Beryllium	beta-Hexachlorocyclohexane	Bis(2-ethylhexyl)
4	2.5	0.047	0.05	0.05	0.65	35	0.071	0.078	4.2
4.9	2.6	0.063	0.24	0.06	1.3	64	0.0883	0.164	4.5
5.5	3.2	0.068	0.35	0.06	1.4	73	0.102	0.195	4.8
5.7	3.6	0.069	0.47	0.06	1.4	82	0.135	0.207	5
5.8	3.6	0.069	0.48	0.07	1.52	110	0.147	0.213	5.3
6	4	0.07	0.56	0.07	1.54	110	0.147	0.282	6.9
6	4	0.091	0.59	0.07	1.6	140.851064	0.153	0.313	7.9
6.4	4	0.091	0.62	0.07	1.72	162.337662	0.15422	0.328	9.9
6.6	4	0.093	0.73	0.07	1.8	214.317673	0.16	0.329	11
6.6	4.1	0.095	0.77	0.07	1.89	250	0.162	0.354	13
7.1	4.1	0.095	0.81	0.07	1.9	250	0.188	0.357	13

The [Distributions] worksheet in the FPMCalc.xls spreadsheet

	Floating Percentile Calculations									
	Page 4: Distribution of Analytical Data Expressed in Percentiles									
Percentiles	4-Methylphenol	Acetone	alpha-Hexachlorocyclohexane	Ammonia	Antimony	Arsenic	Benzoic acid	Beryllium	beta-Hexachlorocyclohexane	Bi
1	5.15	2.58	0.06	0.48	0.06	1.56	47.47	0.08	0.19	
2	5.67	3.01	0.07	0.72	0.07	1.90	59.94	0.10	0.21	
3	5.85	3.41	0.07	0.91	0.07	2.04	66.61	0.11	0.29	
4	6.00	3.60	0.07	1.19	0.08	2.12	70.48	0.14	0.33	
5	6.42	3.68	0.07	1.49	0.08	2.27	74.35	0.14	0.35	
6	6.60	4.00	0.07	1.60	0.09	2.36	78.22	0.15	0.36	
7	7.07	4.00	0.07	1.76	0.09	2.40	82.28	0.15	0.39	
8	7.37	4.00	0.08	2.18	0.09	2.52	94.32	0.15	0.42	
9	7.66	4.00	0.09	2.37	0.09	2.58	106.36	0.15	0.43	
10	8.44	4.04	0.09	2.69	0.09	2.63	110.00	0.16	0.44	
11	8.66	4.10	0.09	2.91	0.10	2.70	110.00	0.16	0.46	
12	8.70	4.11	0.09	3.16	0.10	2.72	111.01	0.17	0.47	

The [Percentiles] worksheet in the FPMCalc.xls spreadsheet

Floating Percentile Calculations

Page 5: Theoretical Hits, No-Hits, and Reliabilities for Each Percentile of the Data Distribution

Percentiles	True Hits	False No-Hits	True No-Hits	False Hits	%False Negatives	%False Positives	Hit Reliability	NoHit Reliability	PredHit Reliability	PredNoHit Reliability	False PredNoHits
1	67	0	0	501	0.0	100.0	100.0	0.0	11.8	NA	NA
2	67	0	0	501	0.0	100.0	100.0	0.0	11.8	NA	NA
3	67	0	0	501	0.0	100.0	100.0	0.0	11.8	NA	NA
4	67	0	0	501	0.0	100.0	100.0	0.0	11.8	NA	NA
5	67	0	0	501	0.0	100.0	100.0	0.0	11.8	NA	NA
6	67	0	0	501	0.0	100.0	100.0	0.0	11.8	NA	NA
7	67	0	0	501	0.0	100.0	100.0	0.0	11.8	NA	NA
8	67	0	0	501	0.0	100.0	100.0	0.0	11.8	NA	NA
9	67	0	0	501	0.0	100.0	100.0	0.0	11.8	NA	NA
10	67	0	0	501	0.0	100.0	100.0	0.0	11.8	NA	NA
11	67	0	0	501	0.0	100.0	100.0	0.0	11.8	NA	NA

The [ErrorCalc] worksheet in the FPMCalc.xls spreadsheet

After setting the initial concentrations, the program finds the analyte with the highest number of False Positives (#FPs), raises its concentration by one increment, and recalculates the %FN and #FP. If the %FN does not exceed the target and the #FPs does not equal zero, the concentration is raised by another increment and retested. If the %FN exceeds the target or the #FPs equals zero, the increment is made smaller and the concentration lowered. This continues until the increment is sufficiently small and a concentration is reached where either (1) the %FN would exceed the desired level if the concentration were raised another increment, or (2) the #FP just reaches 0 and would go up again if the concentration were lowered another increment.

The Percent Precision setting on the [ControlScreen] page controls how small you want the increment to become before ending the calculations. It is roughly a comparison of the final size of the increment to the smallest concentration present for the analyte being tested. The lower you set the precision, the smaller the increment must become before the calculations stop. You may want to experiment with this setting but for consistency with the RSET FPM calculations we recommend that you leave the "Percent Precision" set to 10.

During the data analysis you have the option of pre-screening the Apparent Effects Threshold concentrations (AETs). If you select this option on the [ControlScreen] page, before starting to raise concentrations one increment at a time the program tests the AET concentration of each analyte to see if it can be used without exceeding the target %FN. If so, the AET is used as the result for that analyte. If not, that analyte is taken through the full FPM calculations one increment at a time. It is NOT recommended that you use this option at this time, as the FPM was not originally designed to run with pre-screening for AETs and the effect of this approach on the final result is unknown, even though it does substantially speed up the calculation process.

To use the AET pre-screening option you also must indicate if you would like the program to test for and omit outliers during the screening. In this program an outlier is defined as a concentration that exceeds the next closest concentration by more than a multiple that you specify. For example, the original AET calculations used an outlier multiplier of 3. If the highest concentration for a given analyte was 15 mg/kg, then that data point would be an outlier if the next highest concentration was less than 5 mg/kg. Although outliers can be omitted when selecting AETs, the outliers remain in the data set and are included in the calculations for %FN and #FP.

When the program runs you can see what stage it is in by looking at the Status Bar at the bottom left of the screen. If you prefer to observe the calculations you can do so by marking "Y" in the "Watch Calculations?" box on the [ControlScreen] page. Watching the calculations will slow the program.

After filling in all of the options on the [ControlScreen] page, click on the "Calculate Floating Percentiles" button to start the calculations. When the program starts it will check for certain errors or omissions. If one is encountered the program will terminate and a message box will appear on the screen telling you what is wrong. When the calculations are complete a message indicating the successful completion will appear on the computer screen. Calculations may take between 10 minutes and 2 hours, depending on the size and complexity of your data set, the options you have chosen, and the processing speed of your computer.

When the calculations have been completed the results are found on the [Criteria] worksheet (shown below). The [Criteria] worksheet has three distinct sections. The uppermost rows are the results section. The first column lists each Target %FNs that was used. The next seven columns list the percentage of positives and negatives along with five other performance parameters used to evaluate the results.

Each of these terms is defined below.

Percent False Negatives = $100 * \text{FalseNoHits} / (\text{TrueHits} + \text{FalseNoHits})$

Percent False Positives = $100 * \text{FalseHits} / (\text{TrueNoHits} + \text{FalseHits})$

Hit Reliability = $100 * \text{TrueHits} / (\text{TrueHits} + \text{FalseNoHits})$

No-Hit Reliability = $100 * \text{TrueNoHits} / (\text{TrueNoHits} + \text{FalseHits})$

Predicted Hit Reliability = $100 * \text{TrueHits} / (\text{TrueHits} + \text{FalseHits})$

Predicted No-Hit Reliability = $100 * \text{TrueNoHits} / (\text{TrueNoHits} + \text{FalseNoHits})$

Overall Reliability = $100 * (\text{TrueHits} + \text{TrueNoHits}) / \text{Total No. of Samples}$

The remaining columns hold the results for each analyte listed in alphabetical order.

Floating Percentile Calculations								Definitions of Reliability Measures	
								Hide Additional Calculation Details	
Page 6: Floating Percentile Calculation Targets and Results								Description of Calculation Details	
Nominal Percentiles	%False Negatives	%False Positives	Hit Reliability	NoHit Reliability	PredHit Reliability	PredNoHit Reliability	Overall Reliability	4-Methylphenol	Ammonia
	1.6	65.3	98.4	34.7	42.6	97.7	55.7	710	775
								2	0
2	1.6	41.9	98.4	58.1	53.6	98.6	71.4	710	775
								2	0
								-	+
	1.6	41.9	98.4	58.1	53.6	98.6	71.4	710	775
								2	0
5	4.9	32.3	95.1	67.7	59.2	96.6	76.8	710	775
								2	0
								-	+
	4.9	32.3	95.1	67.7	59.2	96.6	76.8	710	775
								2	0
10	9.8	28.2	90.2	71.8	61.1	93.7	77.8	710	775
								2	0
								-	+

The results section of the [Criteria] worksheet in the FPMCalc.xls spreadsheet

The results are displayed in 5-row sets. Under the performance measures, the first row contains the results for the initial set of concentrations at each selected %FN and the third row contains the results for the final set of concentrations. Under the analyte names:

Row 1 contains the initial concentrations determined by the allowable %FN,

Row 2 contains the #FP for each of the initial concentrations,

Row 3 contains the final concentrations determined by the FPM,

Row 4 contains the #FP for each of the final concentrations, and

Row 5 contains a code indicating which of the two limiting criteria determined each final concentration.

A code of “+” means the final concentration is where #FP = 0. A code of “-” means that increasing the concentration would exceed the desired %FN.

If you click on the “Hide Additional Calculation Details” button the results section table will collapse so that only the final concentrations in the third row of each group of five rows is displayed. This will allow you to more easily compare the results for different %FNs.

Floating Percentile Calculations								Definitions of Reliability Measures	
								Show Additional Calculation Details	
Page 6: Floating Percentile Calculation Targets and Results								Description of Calculation Details	
Nominal Percentiles	%False Negatives	%False Positives	Hit Reliability	NoHit Reliability	PredHit Reliability	PredNoHit Reliability	Overall Reliability	4-Methylphenol	Ammonia
2	1.6	41.9	98.4	58.1	53.6	98.6	71.4	710	775
5	4.9	32.3	95.1	67.7	59.2	96.6	76.8	710	775
10	9.8	28.2	90.2	71.8	61.1	93.7	77.8	710	775

To restore the hidden rows from the results section click on the “Show Additional Calculation Details” button.

The other two buttons at the top of this page can be used to display the forms shown in Section 2.4.3.

The data summary section of the [Criteria] worksheet is located below the results section.

Nominal Percentiles	%False Negatives	%False Positives	Hit Reliability	NoHit Reliability	PredHit Reliability	PredNoHit Reliability	Overall Reliability	4-Methylphenol	Ammonia
	Number of Data Points = Hits							11	44
	Number of Data Points = Indeterminates							0	0
	Number of Data Points = No Hits							33	84
	Total Number of Data Points							44	128
	Minimum concentration							8.4	0.24
	Maximum concentration							6310	775
	AET							2360	775
	Number of Outliers Screened Out								
	AET Note: '>' means largest NH exceeds largest H; 'No NH' means no No-Hit values in data set.								
	Interim 'SortOrder' values stored here								
	Number of iterations used for this calculation							3	4
	Estimated number of iterations for specified precision							3	4
	Increment used for most recent iteration							630.16	77.476
	First Pass Results							710	224

The data summary section of the [Criteria] worksheet in the FPMCalc.xls spreadsheet

This is where basic information about the current data set, like the number of data points, maximum and minimum concentrations, etc. are stored. Current values of data used during the calculations, like the size of the increments, are also stored on this part of the worksheet.

The bottom of the [Criteria] worksheet contains the manual test section.

The area below is reserved for manually checking FPM calculations								
Count Hits/NoHits in Test Row	True Pred Hits	True Pred NH	False Pred Hits	False Pred NH	Hit Stations	NH Stations	Total Stations	False positive counts are stored in
	58	63	61	3	61	124	185	2 0
	% False Neg	% False Pos	% Sensitivity	% Efficiency	% PredHit Rel	% PredNoHit Rel	% Reliability	Place concentrations to be tested
	4.9	49.2	95.1	50.8	48.7	95.5	65.4	710 775
Clear This Table	Hit/NoHit	Survey	Station	Sample	Correct	False Neg	False Pos	4-Methylphenol Ammonia
	0	LKUNION	8	8237	0	0	2	
	1	LKUNION	9	8238	1	0	0	
	1	LKUNION	11.00	8240	1	0	0	
	1	LKUNION	15	8244	1	0	0	
	0	LKUNION	18	8247	0	0	2	
	1	LKUNION	19	8248	1	0	0	
	0	LKUNION	20	8249	1	0	0	
	0	LKUNION	21	8250	1	0	0	
	1	LUUCS000	527	L17656-18	1	0	0	
	1	LUUCS000	535	L17656-6	1	0	0	

The manual test section of the [Criteria] worksheet in the FPMCalc.xls spreadsheet

You can place any set of concentrations that you want to test against the current data set into the row of white cells labeled “Place concentrations to be tested manually in the row below” and then click on the “Count Hits/NoHits in Test Row” button. The number of true and false hits and no-hits along with the various performance measures for those concentrations will be displayed in the yellow rows next to the “Count...” button. If you paste a set of calculated results from the results section into the test row, you can make changes to some of the concentrations and see if the new values make the results better or worse.

Reminder: As noted earlier in this document, if you plan to copy data from one place in the spreadsheet and paste it into another location you should use the “Paste Special” option in the “Edit” menu and select “Paste Values.” If you use the regular paste function you may accidentally paste formulas into the cells instead of their numerical values. Also, the regular paste function may change the formats of the pasted cells to the formats of the copied cells.

An additional worksheet, [DataStorage], is provided in case you want to save the results from previous calculations. This page is only for storage; no calculations take place in [DataStorage].

3. Spreadsheet Macros

3.1 Introduction

3.1.1 Document Text Conventions

In the sections where the macros are discussed the text of the macros and explanatory text included in the body of the macros is displayed in 9-point Courier New font. The macro name and brief explanatory text are shown in bold to make it easier to find the start of each macro. For example:

```
' *****  
' Sub Auto_Open(): Sets up spreadsheet toolbars, menus, etc.  
' Works automatically on opening.  
' *****  
  
Sub Auto_Open()  
    Application.ScreenUpdating = False  
    Call SetUpSheets  
    Call SaveToolbars  
    Call AddNewMenuItem  
    Call ViewTitle  
    Application.ScreenUpdating = True  
End Sub
```

Many of the macros already include explanatory text in the body of the code. Each line of this text is always preceded by an apostrophe. For example:

```
' Make sure that the data in the ChemData and BioData tables are  
' listed in the same order so comparisons can be made. This sorts each  
' table into ascending order by survey, station, and sample.
```

Additional explanatory text not found in the macros is shown in a box in 9 point Arial font. For example:

In the MakeDataTable Module, NumColumnSort values were assigned ...

3.1.2 Common Naming and Macro Conventions

- In designing each worksheet, the Excel "Insert/Name" tool was used to name certain individual cells, part or all of certain rows or columns, or a rectangular table of cells. For example, there is a single value for the "Number of Samples" on the [DataSetSummary] page of the FPMData spreadsheet. "NumSamples" is the name given to the cell containing the value for the number of samples. Therefore, when you want to refer to this cell in a macro you only have to use "NumSamples" and do not have to know what row or column the cell is in. Quite a few of the data are presented as groups listed in columns in data tables. For example, in the "Initial Analytical Data" table on the [InputData] page there are seven columns of data, Survey, Station, Sample, etc., each with room for 65,000 data. "Conc" is the name given to 65,000 cells in the column storing the concentration data. In order to use "Conc" in a macro equation you must specify which concentration you want to use. This is accomplished by attaching an index to the name, such as Conc(1) or Conc(23). Conc(1) is the first entry in the 65,000-cell column named "Conc" and Conc(23) is the 23rd entry. When writing equations for groups of variables like "Conc" values, "i," "j" or "k" was usually used as the index. Therefore, expressions like "Conc(i)" are used when names in equations stand for more than a single value.

- Though not necessary, for convenience and reference all names used in a specific spreadsheet were stored on a hidden worksheet page entitled [Names]. Some needed pieces of information are also stored on the [Names] worksheet, such as the names of the menu bars that are closed when the program opens, thus allowing the program to restore the original menus when the program closes.
- Each of the modules has been designated “Option Explicit.” This is a more tedious process because it requires you to declare all variables that you plan to use before you use them. This is very helpful, however, to avoid such common mistakes as misspelling variables or using them incorrectly. If a variable is used only in the module in which it is declared, “dimension” (Dim) statements were often used at the beginning of the module or at the beginning of the subroutine to declare them. For example:

```
Dim Time As Range, Temp As Single
```

Each of the modules has also been designated as an “Option Private Module.” Therefore, if you want to use variables in modules other than the one in which they are declared you must use the “Public” statement to declare them. For example:

```
Public DataTable As Range, DataUnits As Range
Public i As Integer, j As Integer, k As Integer
```

- Before you can refer to a variable in a macro equation you must first initialize the name by setting it equal to a specified cell or range of cells that you already have declared as mentioned above.

```
Set Time = Range("Time")
Set DataTable = Range("DataTable")
```

The names you plan to use in a macro do not have to be exactly the same as those used for the spreadsheet ranges. More often than not this was done for convenience. (You do NOT have to do this for all of the names you have created in the spreadsheet, just for the names that you intend to use in the macros.)

- Most of the macros that affect what is seen on the screen are written with the following opening and closing lines:

```
Application.ScreenUpdating = False

(Lines of macro code)

Application.ScreenUpdating = True
```

This is to avoid all of the visual “chatter” that occurs on the screen if the spreadsheet must stop and update each cell after each calculation. It also speeds up the overall process of calculation.

- Although not necessary, the term “Call” was used when running a macro from inside another macro. That makes it easier to see what other macros are used and follow the overall process. For example:

```
Sub Auto_Open()
    Application.ScreenUpdating = False
    Call SetUpSheets
    ...
End Sub
```

3.1.3 Macro Usage

Some of the spreadsheets macros run automatically when you open or close the spreadsheet. Others are activated when you click on buttons located on some of the worksheets. Most of the macros function as parts of larger processes and are run as needed ("called") from within one of the automatic or button-activated macros. Some of these are used to format the data and the worksheets, and are not necessary to obtain the final results.

All three spreadsheets have macros named `SetRanges()` and `RunProgram()`, which are located in the `mod01RunProgram` module. These macros do not have identical code in each of the spreadsheets but they serve identical purposes.

3.1.3.1 The `SetRanges()` Macro

The `SetRanges()` macro initializes the variables that are used in the program. If the variables are not initialized, any attempt to use them will result in an error. If the program encounters an error and the execution of the program has to be stopped, the variables will have to be initialized again before they can be used when the program is restarted. Therefore, the macro `SetRanges()` is called from several of the other macros to ensure that the variables are ready to be used regardless of where you start or restart the program.

3.1.3.2 The `RunProgram()` Macro

The `RunProgram()` macro carries out the predominant functions of the spreadsheet. On each spreadsheet it is activated by a button that is labeled with red letters. Most of the work of the `RunProgram()` macro is carried out by other macros that are called from this macro. For this reason the `RunProgram()` macro code can be viewed like an outline that summarizes and displays the order of the program activities.

3.2 Macro Code for FPMData.xls

3.2.1 Summary

FPMData.xls contains 28 Microsoft Excel® Visual Basic macros that perform specific functions for the spreadsheet. The macros are stored in six modules. The code from each module along with some explanatory text is listed in Sections 3.2.2 to 3.2.7. A password is required to access the modules in the spreadsheet.

3.2.1.1 Macros Activated When Spreadsheet Opens or Closes

MacroName() Module Containing Macro	Macro Action
Auto_Open() mod06FormatData	When you open the spreadsheet this macro (1) calls the macro SetRanges() in mod01RunProgram to initialize all of the variables, (2) calls the macro SetUpScreen() in mod06FormatData to close unneeded toolbars, and (3) resets all worksheets to the standard top-left orientation.
Auto_Close() mod06FormatData	When you close the spreadsheet this macro restores the toolbars that were closed when the program opened and resets the formats of the worksheets.

3.2.1.2 Macros Activated from Buttons on Worksheets

MacroName() Module Containing Macro	Button that Activates Macro [Worksheet with Button]	Macro Action
ClearInitialAnalyticalData() mod06FormatData	Clear Analytical Data [InputData]	Asks to make sure you want to delete the data. If so, it deletes all entries from the Initial Analytical Data Table on the [InputData] worksheet.
ClearAllTables() mod06FormatData	Clear Previous [InputData]	Deletes entries from all tables except [InputData]. Calls the macros ClearRawAndDataTables() and ClearSumLists(), both of which are in mod06FormatData.
StripBlanks() mod06FormatData	Remove Blanks [InputData]	Removes leading and trailing blanks from the text entries in the Initial Analytical Data table.
DupeCheck() mod01RunProgram	Check for Duplicates [InputData]	Uses SortInputData() macro in mod06FormatData to sort data by survey, station, sample, and analyte. Then looks for consecutive entries with identical values for each of those fields.
CodeKey() mod06FormatData	Click to View Data Codes [InputData]	Displays a userform that explains the codes in the Screened Data Codes table. See Section 2.1.3.1
GenerateLists() mod05MakeLists	Generate Data Lists [OmitAnalytes]	Uses ChemList() and DataQualList() macros in mod05MakeLists to fill the lists on the [OmitAnalytes] worksheet.
SumsChemList() mod05MakeLists	Create Analyte List [CreateSums]	Creates and formats a list of analytes on the [CreateSums] worksheet.

MacroName() Module Containing Macro	Button that Activates Macro [Worksheet with Button]	Macro Action
NA	<input type="checkbox"/> Retain Group List [CreateSums]	Although this may look like a button that activates a macro, it is <u>not</u> . This is a checkbox used to prevent the current group information from being deleted by the Clear Previous button on [InputData].
ClearSumLists() mod06FormatData	Clear Lists [CreateSums]	Deletes entries from tables on the [CreateSums] worksheet, restores the default color (gray) to the cells, and resets the font color to black.
RunProgram() mod01RunProgram	Create Data Table [CreateSums]	See Section 3.2.1.3
ColorKey() mod06FormatData	Click Here to View Keys to Cell Colors [AnalyteSummary]	Displays a userform that explains the colors used in cells on the [AnalyteSummary] page. See Section 2.1.3.2
SaveGroup() mod06FormatData	Save Group [DataSetSummary]	Saves the current group information to the FPMDataGroups.xls file.
RetrieveGroup() mod06FormatData	Retrieve Group [DataSetSummary]	Retrieves previously saved group information from the FPMDataGroups.xls file.

3.2.1.3 Macros that Control the Main Function of the Program

When you click on the “**Create Data Table**” button, which is located on the [CreateSums] worksheet, you start the RunProgram() macro in the mod01RunProgram module. This macro performs some basic functions on its own, but its primary function is to run a series of macros that each carry out part of the overall process. In the table below the RunProgram() macro code is shown in the left column and the steps that are carried out are explained in the right column.

RunProgram() Macro Code	Function
Dim Time As Range, Temp As Single	Creates a variable in which to store the overall program run time (Time) and a variable used to temporarily store the starting time (Temp).
Set Time = Range("Time") Time.ClearContents Temp = Timer	Initializes the Time variable and stores it in a cell named “Time.” Deletes anything currently in that cell and stores the current time (Timer), which will be used as the program start time.
Application.StatusBar = "(1/4) Clearing Tables and Checking Selected Inputs"	Puts a message in the status bar (located in the lower left corner of the spreadsheet screen) to inform the user that the program is beginning the first of four steps and describes what will be done in that step.
Range("A1").Select Call SetRanges ThisFileName = "FPMData.xls Version 122808"	Stores the cursor off-screen (column A and row 1 are hidden on every worksheet). Calls SetRanges() in mod01RunProgram to initialize the variables used by the program. (See Section 3.1.3.1 for more information about SetRanges()). Places current version name on the [InputData] page.

RunProgram() Macro Code	Function
DataSummary.ClearContents NameScreenedAnalytes.ClearContents NumColumnSort.ClearContents ScreenedOut.ClearContents Call ClearRawAndDataTables	Deletes old entries from some of the tables in preparation for making new tables. Calls ClearRawAndDataTables() in mod06FormatData to delete entries from additional tables.
Application.ScreenUpdating = False	Prevents the program from updating the screen after each step. This allows the program to run faster.
NumChemInitial = Application.CountA(AnalyteNameList) NumChemSummed = Application.CountA(SumListNums) NumNewGroups = Application.CountA(SumGroupNames) NumDataInitial = Application.CountA(InputName) NumQual = Application.CountA(QualList)	Evaluates some of the parameters that will be needed in later procedures: NumChemInitial = initial no. of analytes in the data set, NumChemSummed = no. of analytes in new groups, NumNewGroups = no. of new groups created, NumDataInitial = no. of data listed on [InputData], and NumQual = no. of data qualifiers used in the data set.
Call InputCheck	Calls InputCheck() in mod01RunProgram to check the validity of some of the numbers that you put in the spreadsheet. If an unacceptable value is found a message will be displayed (see example). The program will end when you click on "OK." <div data-bbox="800 942 1372 1127" data-label="Image"> </div>
Application.StatusBar = "(2/4) Compiling Raw Data"	Updates the note in the status bar.
Call CompileRawData	Calls CompileRawData() in the mod02CompileRawData, which does the following: (1) Puts analyte names into table headers. If you have chosen to omit an analyte it colors the cell with that name gray. It then counts how many data points will be eliminated for that analyte and identifies it by putting "Analyte" in the first column of the ScreenedData table on the [InputData] page. (2) Sorts initial data in order of sample, survey, and station; and counts the number of different samples in the data set. (3) If a data point has a data qualifier that you want to omit, it identifies it by putting "DQ" in the second column of the ScreenedData table on the [InputData] page. On the [AnalyteSummary] page it colors the cell containing that data point red.
Application.StatusBar = "(3/4) Summarizing Raw Data"	Updates the note in the status bar.
Call SummarizeData	Calls SummarizeData() macro in the mod03SummarizeData, which does the following: (1) Writes names and units of new groups in table on the [AnalyteSummary] page.

RunProgram() Macro Code	Function
	<p>(2) Adds the data points for all group members to generate a total for the group. Data points marked "DQ" as noted above are not included in the sum.</p> <p>(3) Identifies the analytes or data points as belonging to one of the following six groups –</p> <ul style="list-style-type: none"> • analytes with sufficient data that are not in a group, • analytes you retain even though they are part of a group, • groups that you created that have sufficient data, • analytes omitted because they are part of a group, • analytes that you choose to omit, and • analytes or sums that have too few data. <p>(4) Colors cells in the [AnalyteSummary] table based on the group to which the analytes or data points belong.</p> <p>(5) Counts the number of data points for each analyte and determines the min and max concentrations. If analyte has too few data, identifies it by putting "Too Few" in the third column of the ScreenedData table on the [InputData] page.</p> <p>(6) Sorts the [AnalyteSummary] table columns in the order of the six groups noted above.</p>
Application.StatusBar = "(4/4) Assembling Final Data Table"	Updates the note in the status bar.
Call MakeDataTable	<p>Calls MakeDataTable() in mod04MakeDataTable, which does the following:</p> <p>(1) Copies the information from [AnalyteSummary] to [DataTable] leaving out data points or analytes marked for omission in the SumarizeData step.</p> <p>(2) Counts the number of analytes in each group.</p> <p>(3) Checks to make sure that the sum of analytes and data points in the final table agrees with the initial number taking into account any new groups or deleted analytes or data points. If there is an error it calls ShowMsg() in mod01RunProgram to display an error message and end the program.</p>
Call DataFormat	<p>Calls DataFormat in mod06FormatData, which does the following:</p> <p>(1) Puts [InputData] back into ascending order by Survey, Station, and Sample.</p> <p>(2) Formats the data tables so that the columns are an appropriate width to make them easier to read.</p> <p>(3) Calls CenterText() in mod06FormatData to center the data in the cells of the various tables.</p>
If RetainGroupList = "False" Then Call GroupList	If you are not reusing existing group names, this calls GroupList in mod05MakeLists, which creates a list of all of the newly-defined groups and all of the analytes that make up each group. This list is on the [DataSetSummary] page.

RunProgram() Macro Code	Function
Call OmitList	Calls OmitList in mod05MakeLists, which creates a list of all analytes and groups that have been screened out. This list is on the [DataSetSummary] page.
Sheets("AnalyteSummary").Select Application.StatusBar = False Application.ScreenUpdating = True Time = Timer – Temp MsgBox (" Calculations Completed ")	Selects the page that will appear when the program ends. Deletes the message from the status bar. Restores screenupdating so changes made during the running of the program will now show up on screen. Subtracts the initial time (Temp) from the current time (Timer) to get the program run time. Puts message on screen to inform the user that the program has ended.

3.2.2 mod01RunProgram

This module contains 5 macros:

RunProgram calls each main subroutine, adds notes to the Status Bar to show progress of the calculations, and times the calculations.

SetRanges initializes the variables by setting them equal to the previously-declared ranges.

InputCheck makes sure that required inputs with appropriate values have been entered into some of the key input ranges before starting to sort and screen the data.

ShowMsg displays error messages generated from the InputCheck macro and shuts the program down after the message has been reviewed.

DupeCheck checks the initial data set to determine if there are any pairs of data with an identical set of values for survey, station, sample, and analyte name.

The code and comments for this module are shown below.

```
' File: FPMData.xls

' This is the first of three spreadsheets used for implementing the
' Floating Percentile Method (FPM) developed by Teresa Michelsen of
' Avocet Consulting. The FPM is used to assess sediment toxicity
' and chemical composition data to develop sediment quality guidelines.

' FPMData.xls:
' (1) accepts up to 65,000 analytical results for up to 250 chemical constituents having
' up to 100 different data qualifiers,
' (2) sorts the raw data into groups by chemical constituent,
' (3) summarizes the data for the data set as a whole and for each constituent,
' (4) generates a final data set with options to
' (a) screen out data points with specified data qualifiers,
' (b) screen out constituents with less than a specified number of data,
' (c) delete specified constituents by name, and
' (d) create up to 25 new analyte groups and sum the data for any number of
' individual constituents.

' This spreadsheet was developed by Michael R. Anderson,
' Oregon Department of Environmental Quality.
```

```

' Last edited: December 28, 2008

' Names shown in [Square Brackets] in the comments are names of worksheets.

Option Private Module
Option Explicit

' *****
' Most variables are declared below as Public variables since they are
' used in more than one subroutine in a given module or in more than one
' module. Some variables used only in one routine are declared in that routine.
' *****

Public AnalyteNameList As Range, AnalyteNameUnits As Range
Public AnalyteOmitState As Range, AnalyteRetainState As Range
Public DataHeader As Range, DataSummary As Range
Public DataTable As Range, DataUnits As Range
Public DupeConc As Range, DupeName As Range, DupeQual As Range
Public DupeSample As Range, DupeStation As Range, DupeSurvey As Range
Public DupeTable As Range, DupeTitle As Range, DupeUnits As Range
Public FinalDataTable As Range, FullDataTable As Range
Public FullInputTable As Range, FullRawDataTable As Range
Public FullRawDataTableNoHeader As Range, FullSummaryTable As Range
Public GroupMemberList As Range, GroupMemberRetain As Range, GroupNameUnits As Range
Public IDTable As Range, InputConc As Range, InputHeader As Range
Public InputName As Range, InputQual As Range, InputSample As Range
Public InputStation As Range, InputSurvey As Range
Public InputTable As Range, InputUnits As Range
Public MaxConc As Range, MinConc As Range
Public MinNumDetects As Range, NameScreenedAnalytes As Range
Public NetNumChemInitial As Range, NumChemOmitted As Range
Public NetNumDataInitial As Range
Public NumChemInitial As Range, NumNewGroups As Range
Public NumChemSumOmitted As Range, NumChemSumRetained As Range
Public NumChemTooFewData As Range, NumColumnSort As Range
Public NumDataBadDQs As Range, NumDataPoints As Range
Public NumDataInitial As Range, NumDataNewGroups As Range
Public NumDataNewGroupsOmitted As Range, NumDataNewGroupsRetained As Range
Public NumDataNewSums As Range, NumDataOmitted As Range
Public NumFinalDataPoints As Range, NumDataTooFewData As Range
Public NumDataSumOmitted As Range, NumDataSumRetained As Range
Public NumNewGroupsRetained As Range, NumNewGroupsOmitted As Range
Public NumOmitDQs As Range, NumSamples As Range, QualList As Range
Public QualState As Range, RawDataHeader As Range
Public RawDataTable As Range, RawDataTableSort As Range, RawDataUnits As Range
Public RawIDTable As Range, RawSample As Range
Public RawStation As Range, RawSurvey As Range
Public RetainGroupList As Range, ScreenedOut As Range
Public ScreenedOut1 As Range, ScreenedOut2 As Range
Public ScreenedOut3 As Range, SumGroupNames As Range
Public SummaryHeader As Range, SumListNames As Range
Public SumListNums As Range, SumListUnits As Range
Public ThisFileName As Range, ToolBarStorage As Range

Public i As Integer, j As Integer, k As Integer, m As Integer, n As Integer
Public ListCapacity As Integer, NumChemSummed As Integer
Public NumCol1 As Integer, NumCol2 As Integer, NumDupes As Integer
Public NumStations As Integer, NumQual As Integer

Public Workrange As Object

Public Comment As String

```

```

' *****
' RunProgram() calls each main subroutine, adds notes to Status Bar
' to show progress of the calculations, and times the calculations.
' *****

Sub RunProgram()

Dim Time As Range, Temp As Single

Set Time = Range("Time")
Time.ClearContents
Temp = Timer

Application.StatusBar = "(1/4) Clearing Tables and Checking Selected Inputs"
Range("A1").Select

Call SetRanges

' When the program is revised change the date in the line below.

ThisFileName = "FPMDData.xls Version 122808"

' Delete old entries from tables in preparation for making new tables.

DataSummary.ClearContents
NameScreenedAnalytes.ClearContents
NumColumnSort.ClearContents
ScreenedOut.ClearContents
Call ClearRawAndDataTables

' This prevents the program from updating the screen after each step,
' which allows the program to run faster.

Application.ScreenUpdating = False

' Evaluate some parameters that will be needed in later procedures:
' NumChemInitial = initial number of analytes in the data set,
' NumChemSummed = no. of analytes incorporated into new groups,
' NumNewGroups = no. of new groups created,
' NumDataInitial = no. of data points listed on the [InputData] worksheet, and
' NumQual = no. of data qualifiers used in the data set.

NumChemInitial = Application.CountA(AnalyteNameList)
NumChemSummed = Application.CountA(SumListNums)
NumNewGroups = Application.CountA(SumGroupNames)
NumDataInitial = Application.CountA(InputName)
NumQual = Application.CountA(QualList)

' Check the validity of selected inputs before screening.

Call InputCheck

' Place note in status bar so that progress of program can be observed.

Application.StatusBar = "(2/4) Compiling Raw Data"
Call CompileRawData

Application.StatusBar = "(3/4) Summarizing Raw Data"
Call SummarizeData

```

```

Application.StatusBar = "(4/4) Assembling Final Data Table"
Call MakeDataTable
Call DataFormat

If RetainGroupList = "False" Then Call GroupList
Call OmitList

Sheets("AnalyteSummary").Select

Application.StatusBar = False
Application.ScreenUpdating = True

Time = Timer - Temp
MsgBox ("    Calculations Completed    ")

End Sub

' *****
' SetRanges() initializes the variables by setting them equal to
' the values in the previously-declared ranges. This is done here
' for variables that are used in more than one macro. Variables
' used in only one macro are initialized in the relevant macro.
' *****

Sub SetRanges()

Set AnalyteNameList = Range("AnalyteNameList")
Set AnalyteNameUnits = Range("AnalyteNameUnits")
Set AnalyteOmitState = Range("AnalyteOmitState")
Set AnalyteRetainState = Range("AnalyteRetainState")
Set DataHeader = Range("DataHeader")
Set DataSummary = Range("DataSummary")
Set DataTable = Range("DataTable")
Set DataUnits = Range("DataUnits")
Set DupeConc = Range("DupeConc")
Set DupeName = Range("DupeName")
Set DupeQual = Range("DupeQual")
Set DupeSample = Range("DupeSample")
Set DupeStation = Range("DupeStation")
Set DupeSurvey = Range("DupeSurvey")
Set DupeTable = Range("DupeTable")
Set DupeTitle = Range("DupeTitle")
Set DupeUnits = Range("DupeUnits")
Set FinalDataTable = Range("FinalDataTable")
Set FullDataTable = Range("FullDataTable")
Set FullInputTable = Range("FullInputTable")
Set FullRawDataTable = Range("FullRawDataTable")
Set FullRawDataTableNoHeader = Range("FullRawDataTableNoHeader")
Set FullSummaryTable = Range("FullSummaryTable")
Set GroupMemberList = Range("GroupMemberList")
Set GroupMemberRetain = Range("GroupMemberRetain")
Set GroupNameUnits = Range("GroupNameUnits")
Set IDTable = Range("IDTable")
Set InputConc = Range("InputConc")
Set InputHeader = Range("InputHeader")
Set InputName = Range("InputName")
Set InputQual = Range("InputQual")
Set InputSample = Range("InputSample")
Set InputStation = Range("InputStation")
Set InputSurvey = Range("InputSurvey")
Set InputTable = Range("InputTable")

```

```

Set InputUnits = Range("InputUnits")
Set MaxConc = Range("MaxConc")
Set MinConc = Range("MinConc")
Set MinNumDetects = Range("MinNumDetects")
Set NameScreenedAnalytes = Range("NameScreenedAnalytes")
Set NetNumChemInitial = Range("NetNumChemInitial")
Set NetNumDataInitial = Range("NetNumDataInitial")
Set NumChemInitial = Range("NumChemInitial")
Set NumChemOmitted = Range("NumChemOmitted")
Set NumChemSumOmitted = Range("NumChemSumOmitted")
Set NumChemSumRetained = Range("NumChemSumRetained")
Set NumChemTooFewData = Range("NumChemTooFewData")
Set NumColumnSort = Range("NumColumnSort")
Set NumDataBadDQs = Range("NumDataBadDQs")
Set NumDataInitial = Range("NumDataInitial")
Set NumDataNewGroups = Range("NumDataNewGroups")
Set NumDataNewGroupsOmitted = Range("NumDataNewGroupsOmitted")
Set NumDataNewGroupsRetained = Range("NumDataNewGroupsRetained")
Set NumDataOmitted = Range("NumDataOmitted")
Set NumDataPoints = Range("NumDataPoints")
Set NumDataSumOmitted = Range("NumDataSumOmitted")
Set NumDataSumRetained = Range("NumDataSumRetained")
Set NumDataTooFewData = Range("NumDataTooFewData")
Set NumFinalDataPoints = Range("NumFinalDataPoints")
Set NumNewGroups = Range("NumNewGroups")
Set NumNewGroupsOmitted = Range("NumNewGroupsOmitted")
Set NumNewGroupsRetained = Range("NumNewGroupsRetained")
Set NumOmitDQs = Range("NumOmitDQs")
Set NumSamples = Range("NumSamples")
Set QualList = Range("QualList")
Set QualState = Range("QualState")
Set RawDataHeader = Range("RawDataHeader")
Set RawDataTable = Range("RawDataTable")
Set RawDataTableSort = Range("RawDataTableSort")
Set RawDataUnits = Range("RawDataUnits")
Set RawIDTable = Range("RawIDTable")
Set RawSample = Range("RawSample")
Set RawStation = Range("RawStation")
Set RawSurvey = Range("RawSurvey")
Set RetainGroupList = Range("RetainGroupList")
Set ScreenedOut = Range("ScreenedOut")
Set ScreenedOut1 = Range("ScreenedOut1")
Set ScreenedOut2 = Range("ScreenedOut2")
Set ScreenedOut3 = Range("ScreenedOut3")
Set SumGroupNames = Range("SumGroupNames")
Set SummaryHeader = Range("SummaryHeader")
Set SumListNames = Range("SumListNames")
Set SumListNums = Range("SumListNums")
Set SumListUnits = Range("SumListUnits")
Set ThisFileName = Range("ThisFileName")
Set ToolBarStorage = Range("ToolBarStorage")

```

```
End Sub
```



```

' *****
' InputCheck() makes sure that required inputs with appropriate
' values have been entered into some of the key input ranges
' before starting to sort and screen data.
' *****

Sub InputCheck()

Dim NumGroups As Integer

For i = 1 To NumChemInitial

' Make sure user is not trying to use an omitted analyte in a sum.

If (AnalyteOmitState(i) = "True" And SumListNums(i) <> "") Then
    Sheets("CreateSums").Select: Range(SumListNames(i), SumListNums(i)).Select
    Comment = "Omitted analytes cannot be used in sums."
    Call ShowMsg
End If

' Make sure user is not trying to retain an analyte that is not used in a sum.

If (AnalyteRetainState(i) = "True" And SumListNums(i) = "") Then
    Sheets("CreateSums").Select: Range(SumListNames(i), SumListNums(i)).Select
    Comment = "You cannot select an analyte to be retained if it is not used in a sum."
    Call ShowMsg
End If

' Make sure user has not marked an analyte both for retention and omission.

If (AnalyteOmitState(i) = "True" And AnalyteRetainState(i) = "True") Then
    Sheets("OmitAnalytes").Select: AnalyteNameList(i).Select
    Comment = "You cannot omit and retain the same analyte (duh)"
    Call ShowMsg
End If

Next i

' Make sure there's an entry in the "Omit Chem w/Data Pts <" box

If (MinNumDetects = "" Or MinNumDetects < 1) Then
    Sheets("OmitAnalytes").Select: MinNumDetects.Select
    Comment = "'Omit Chem w/Data points <' must contain an integer >= 1."
    Call ShowMsg
End If

' Make sure the entry in the "Omit Chem w/Data Pts <" box is an integer

If (MinNumDetects <> Int(MinNumDetects)) Then
    Sheets("OmitAnalytes").Select: MinNumDetects.Select
    Comment = "'Omit Chem w/Data points <' must contain an INTEGER."
    Call ShowMsg
End If

```

```

'   Make sure the analyte list is has been created

If AnalyteNameList(1) = "" Then
    Sheets("OmitAnalytes").Select: AnalyteNameList.Select
    Comment = "'Analyte List' must be populated before data can be screened."
    Call ShowMsg
End If

'   Make sure that the numbers of all defined group names are used in the analyte list,
'   and all of the numbers used in the analyte list are associated with defined group names.

For i = 1 To 25
    If i <= NumNewGroups And SumGroupNames(i) = "" Then
        Comment = "Defined group names must start on the first line and be listed consecutively
without skipping any lines."
        Sheets("CreateSums").Select: SumGroupNames.Select
        Call ShowMsg
    ElseIf i > NumNewGroups And SumGroupNames(i) <> "" Then
        Comment = "There is no group name defined for " & i & "."
        Sheets("CreateSums").Select: SumGroupNames.Select
        Call ShowMsg
    ElseIf i <= NumNewGroups And Application.CountIf(SumListNums, i) < 2 Then
        Comment = "There must be at least two analytes for each defined group number."
        Sheets("CreateSums").Select: SumListNums.Select
        Call ShowMsg
    ElseIf i > NumNewGroups And Application.CountIf(SumListNums, i) > 0 Then
        Comment = "Every number entered into the analyte list must have an associated group name."
        Sheets("CreateSums").Select: SumListNums.Select
        Call ShowMsg
    End If
Next i

'   Make sure that all members of a given group have data with compatible units.

NumGroups = Application.CountA(SumGroupNames)
GroupNameUnits = ""

For i = 1 To NumGroups

    For j = 1 To NumChemInitial
        If SumListNums(j) <> "" Then
            If SumListNums(j) = i And GroupNameUnits(i) = "" Then
                GroupNameUnits(i) = SumListUnits(j)
            ElseIf SumListNums(j) = i And SumListUnits(j) <> GroupNameUnits(i) Then
                Sheets("CreateSums").Select
                Range(SumGroupNames(i), GroupNameUnits(i)).Select
                Comment = "The data for all analytes in a group must have the same units."
                Call ShowMsg
            End If
        End If
    Next j

Next i

End Sub

```

```

' *****
' ShowMsg() Displays relevant error message from InputCheck()
' and MakeDataTable() and shuts the program down after the
' message has been reviewed.
' *****

Sub ShowMsg()

Selection.Interior.Pattern = xlLightUp
Application.ScreenUpdating = True

MsgBox (Comment)

Selection.Interior.Pattern = xlNone
Range("A1").Select
Application.StatusBar = "": End

End Sub

' *****
' DupeCheck tests to see if two or more data points have identical values
' for all four identifiers: survey, station, sample, and analyte name.
' If any are found, one of each pair is moved to a separate storage table.
' This is called from a button on the [InputData] page.
' *****

Sub DupeCheck()

Application.ScreenUpdating = False

Call SetRanges
Call SortInputData

NumDups = 0
NumDataInitial = Application.CountA(InputName)
DupeTable.ClearContents

For i = 1 To NumDataInitial
    If InputSurvey(i) = InputSurvey(i - 1) Then
        If InputStation(i) = InputStation(i - 1) Then
            If InputSample(i) = InputSample(i - 1) Then
                If InputName(i) = InputName(i - 1) Then
                    NumDups = NumDups + 1
                    DupeSurvey(NumDups) = InputSurvey(i): InputSurvey(i) = ""
                    DupeStation(NumDups) = InputStation(i): InputStation(i) = ""
                    DupeSample(NumDups) = InputSample(i): InputSample(i) = ""
                    DupeName(NumDups) = InputName(i): InputName(i) = ""
                    DupeConc(NumDups) = InputConc(i): InputConc(i) = ""
                    DupeUnits(NumDups) = InputUnits(i): InputUnits(i) = ""
                    DupeQual(NumDups) = InputQual(i): InputQual(i) = ""
                End If
            End If
        End If
    End If
Next i

```

```

If NumDuples = 0 Then

    MsgBox ("No duplicates have been found.")

ElseIf NumDuples > 0 Then

    FullInputTable.Sort _
    Key1:=InputSurvey, Order1:=xlAscending, _
    Key2:=InputStation, Order2:=xlAscending, _
    Key3:=InputSample, Order3:=xlAscending, _
    Header:=xlYes, _
    Orientation:=xlTopToBottom

    Application.ScreenUpdating = True

    DupeTitle.Select
    MsgBox (NumDuples & " duplicate(s) have been found.")

End If

End Sub

```

This is the end of FPMDData.xls mod01RunProgram.
--

3.2.3 mod02CompileRawData

This module contains only 1 macro, CompileRawData . The code and comments are shown below.

```
Option Private Module
Option Explicit
```

```
' *****
' CompileRawData() sorts the data on the [InputData] page into groups by analyte
' and stores the results on the [AnalyteSummary] page. It also identifies some
' of the data points that will be screened out before carrying out the Floating
' Percentile calculations.
' *****

Sub CompileRawData()

' Copy the names from "Analyte List" on the [OmitAnalytes] worksheet into the
' header rows on both the Summary Table and the Sorted Data Table on the
' [AnalyteSummary] worksheet.

NumChemOmitted = 0
NumDataInitial = Application.CountA(InputName)
For i = 1 To NumChemInitial

    RawDataHeader(i) = AnalyteNameList(i)
    SummaryHeader(i) = AnalyteNameList(i)
    RawDataUnits(i) = AnalyteNameUnits(i)
    NumOmitDQs(i) = 0

' If the analyte is going to be omitted (i.e., you put a check next to it
' in the Analyte List) identify it by coloring the cell with its name in
' the header rows dark gray (ColorIndex = 16). Also make the font a lighter
' color to make it easier to read against the dark gray background.

    If AnalyteOmitState(i) = "True" Then
        SummaryHeader(i).Interior.ColorIndex = 16
        SummaryHeader(i).Font.ColorIndex = 2
        RawDataHeader(i).Interior.ColorIndex = 16
        RawDataHeader(i).Font.ColorIndex = 2

' Check the data on the [InputData] page and mark each data point
' for the omitted analyte by putting "Analyte" in the first column
' of the ScreenedData table on the [InputData] page.

        For j = 1 To NumDataInitial
            If InputName(j) = AnalyteNameList(i) Then ScreenedOut1(j) = "Analyte"
        Next j

    End If

Next i
```

```

' Sort the [InputData] in ascending order by sample, survey, and
' station in preparation for counting the number of different samples
' in the data set.

FullInputTable.Sort _
    Key1:=InputSample, Order1:=xlAscending, _
    Key2:=InputSurvey, Order2:=xlAscending, _
    Key3:=InputStation, Order3:=xlAscending, _
    Header:=xlYes, _
    Orientation:=xlTopToBottom

' To determine the number of samples, go down the list and count how
' often the sample name changes. Since the data are sorted by sample name,
' each change represents a new sample and the total is the number of
' different samples. Note that the first entry (i.e., InputSample(1))
' is immediately counted because it differs from InputSample(0), which
' is the column name in the header.

NumSamples = 0
For i = 1 To NumDataInitial

' Each time a new sample name is encountered, count it and copy the ID
' information (survey, station, and sample names) from [InputData] to
' the next available row in the Sorted Data Table on the [AnalyteSummary]
' worksheet.

    If InputSample(i) <> InputSample(i - 1) Then
        NumSamples = NumSamples + 1
        RawSurvey(NumSamples) = InputSurvey(i)
        RawStation(NumSamples) = InputStation(i)
        RawSample(NumSamples) = InputSample(i)
    End If

' If the data point has a data qualifier that is to be omitted (i.e., you
' put a check next to it in the Data Qualifier list), put "DQ" in the second
' column of the Screened Data table on the [InputData] page and then count
' it by adding 1 to the number of omitted DQs under the appropriate analyte
' name on the [AnalyteSummary] worksheet.

    For j = 1 To NumQual
        If (InputQual(i) = QualList(j) And QualState(j) = "True") Then
            ScreenedOut2(i) = "DQ"

            For k = 1 To NumChemInitial
                If InputName(i) = RawDataHeader(k) Then
                    NumOmitDQs(k) = NumOmitDQs(k) + 1
                    Exit For
                End If
            Next k

        End If
    Next j

Next i

```



```

' Enter the concentration into the Sorted Data table on the [AnalyteSummary]
' worksheet. First, find the correct column on [AnalyteSummary] by matching
' the chemical name. Then copy the analytical result from [InputData] to
' [AnalyteSummary] and jump out of loop. If this data point is to be omitted
' due to unacceptable DQ, identify it by coloring the cell red (ColorIndex = 3),
' otherwise uncolor the cell (ColorIndex = xlNone).

For j = 1 To NumChemInitial
    If RawDataHeader(j) = InputName(i) Then
        RawDataTable(NumSamples, j) = InputConc(i)
        If ScreenedOut2(i) = "DQ" Then
            RawDataTable(NumSamples, j).Interior.ColorIndex = 3
        Else
            RawDataTable(NumSamples, j).Interior.ColorIndex = xlNone
        End If
    End If
    Exit For
End If
Next j

Next i

End Sub

```

This is the end of FPMData.xls mod02CompileRawData.

3.2.4 mod03SummarizeData

This module contains only 1 macro, **SummarizeData**. The code and comments are shown below.

```
Option Private Module
Option Explicit
```

```
' *****
' SummarizeData() generates columns of data for the new groups of summed
' analytes. Then it summarizes information about all of the analytes, old
' and new, from the [AnalyteSummary] worksheet and stores it in a Summary
' Table on the [AnalyteSummary] worksheet.
' *****

Sub SummarizeData()

' Write the names and units of the new groups in the headers on the [AnalyteSummary]
' page. Start with the first empty column after the original analytes.

For k = 1 To NumNewGroups
    RawDataHeader(NumChemInitial + k) = SumGroupNames(k)
    RawDataUnits(NumChemInitial + k) = GroupNameUnits(k)
    SummaryHeader(NumChemInitial + k) = SumGroupNames(k)
Next k

' To create the new sums, check each analyte at every station.
' If the analyte is not selected for summing, skip it (GoTo Continuel).
' If the analyte is selected but the data point cell is empty, skip it.
' If selected and there is a data point but it doesn't have an OK DQ, skip it.
' If selected, there is a data point, and DQ is OK, add it to the appropriate sum.

For i = 1 To NumSamples
    For j = 1 To NumChemInitial
        If SumListNums(j) = "" Then GoTo Continuel
        If RawDataTable(i, j) = "" Then GoTo Continuel
        If RawDataTable(i, j).Interior.ColorIndex = 3 Then GoTo Continuel

        RawDataTable(i, NumChemInitial + SumListNums(j)) = _
            RawDataTable(i, NumChemInitial + SumListNums(j)) + RawDataTable(i, j)
    Continuel:

        Next j
    Next i

    Sheets("AnalyteSummary").Select

    For k = 1 To NumChemInitial + NumNewGroups
        NumColumnSort(k) = 0
    Next k

' After all of the data have been compiled and, if necessary, summed,
' the raw data table is sorted into the following six groups:

' (1) analytes with sufficient data that you did not put into a group,
' (2) analytes that you choose to retain even though they are part of a group,
' (3) groups that you created that have sufficient data,
' (4) analytes that are omitted by default because they are part of a group,
' (5) analytes that you choose to omit regardless of the amount of data, and
```

```
' (6) analytes or sums that have less than the minimum number of data.
'
' Additional explanation on how the data are handled:
'
' (a) If you choose to omit an analyte, it will be omitted regardless of the number of data.
' You cannot use such an analyte in a group.
' (b) If you put an analyte into a group, only the group will appear in the final data table.
' The analyte will not be shown in the final table unless you choose to retain it.
' (c) If you choose to retain an analyte that is part of a group, it will be retained regardless
' of the number of data. You cannot retain an analyte that you have omitted.
' (d) If an analyte that you have selected to be in a group has fewer than the specified
' minimum number of data, it will still be included in the sum for that group.
' (e) If an analyte has fewer than the specified minimum number of data, it will not
' be included in the final data table unless you have chosen to retain it.
' (f) If a group has fewer than the specified minimum number of data, that group will be
' omitted from the final data table.
'
' To prepare for that sorting step, whenever an analyte is identified as belonging
' to one of the six groups described above, the relevant integer is written above
' it in a hidden row named NumColumnSort.
```

NumColumnSort is a row hidden at the top of the [Analyte Summary] page from F1 to IU1.

```
For k = 1 To NumChemInitial
```

```
' Identify the analytes that have been incorporated into groups (SumListNums(k) <> "").
' If a grouped analyte is to be included in the final table (AnalyteRetainState(k) =
' "True"), color the headers for that analyte on the [AnalyteSummary] worksheet blue
' (ColorIndex = 33) and add 2 to the NumSortColumn. If a grouped analyte is not being
' retained (AnalyteRetainState(k) = "False"), color the headers for that analyte on the
' [AnalyteSummary] worksheet orange (ColorIndex = 45) and add 8 to the NumSortColumn.
```

```
    If SumListNums(k) <> "" And AnalyteRetainState(k) = "True" Then
        SummaryHeader(k).Interior.ColorIndex = 33
        RawDataHeader(k).Interior.ColorIndex = 33
        NumColumnSort(k) = NumColumnSort(k) + 2
```

```
    ElseIf SumListNums(k) <> "" And AnalyteRetainState(k) = "False" Then
        SummaryHeader(k).Interior.ColorIndex = 45
        RawDataHeader(k).Interior.ColorIndex = 45
        NumColumnSort(k) = NumColumnSort(k) + 8
```

```
' Identify the analytes that you want to omit (AnalyteOmitState(k) = "True").
' When one of these is encountered, add 16 to the NumSortColumn. The header
' cells were already colored dark gray in the CompileRawData module.
```

```
    ElseIf AnalyteOmitState(k) = "True" Then
        NumColumnSort(k) = NumColumnSort(k) + 16
```

```
' If an original analyte is not part of a group and is not being omitted,
' identify it by coloring the header light yellow (ColorIndex=36)
' and add 1 to the NumSortColumn.
```

```
    Else
        SummaryHeader(k).Interior.ColorIndex = 36
        RawDataHeader(k).Interior.ColorIndex = 36
        NumColumnSort(k) = NumColumnSort(k) + 1
```

```
    End If
Next k
```

```

' That completes the columns of the original data. Now check the
' columns of newly-formed sums. Identify them by coloring their
' headers green (ColorIndex=50) and add 4 to the NumSortColumn.
' Since the sums are made only from data with acceptable quality,
' place a 0 in the summary table for the "Number with Unacceptable
' Data Qualifiers."

For k = 1 To NumNewGroups
    SummaryHeader(NumChemInitial + k).Interior.ColorIndex = 50
    RawDataHeader(NumChemInitial + k).Interior.ColorIndex = 50
    NumColumnSort(NumChemInitial + k) = NumColumnSort(NumChemInitial + k) + 4
    NumOmitDQs(NumChemInitial + k) = 0
Next k

' Review all old and new analytes and summarize number of data
' points, maximum value, minimum value, etc. for each analyte.

NumChemTooFewData = 0
For i = 1 To NumChemInitial + NumNewGroups

    ' Count number of initial data points, find the maximum & minimum,
    ' and calculate the final number of data points for each analyte.

    Range(RawDataTable(1, i), RawDataTable(NumSamples, i)).Select
    Set Workrange = Selection
    NumDataPoints(i) = Application.CountA(Workrange)
    MinConc(i) = Application.Min(Workrange)
    MaxConc(i) = Application.Max(Workrange)
    NumFinalDataPoints(i) = NumDataPoints(i) - NumOmitDQs(i)

    ' Compare the number of data points to the minimum requirement. If one of the
    ' original analytes does not have enough data, mark the data points on [InputData]
    ' as "Too Few," count the analyte, and add 32 to its sort number. If a group does
    ' not have enough data, just count it and add 32 to its sort number. Then all of
    ' these analytes, even those omitted or retained, will be marked as having too
    ' few data by adding diagonal lines to the header cell.

    If NumFinalDataPoints(i) < MinNumDetects Then
        If i <= NumChemInitial Then
            If AnalyteOmitState(i) = "False" Then
                For k = 1 To NumDataInitial
                    If InputName(k) = RawDataHeader(i) Then ScreenedOut3(k) = "Too Few"
                Next k
                NumColumnSort(i) = NumColumnSort(i) + 32
            End If
        Else
            NumColumnSort(i) = NumColumnSort(i) + 32
        End If

        SummaryHeader(i).Interior.Pattern = xlLightUp
        RawDataHeader(i).Interior.Pattern = xlLightUp

    End If

Next i

```

```
' As discussed above, use the numbers that were just placed in NumColumnSort
' to sort the Sorted Data table on the [AnalyteSummary] worksheet so that the
' analytes are grouped in the following order:
' (1) remaining original analytes (light yellow, "NetNumChemInitial"),
' (2) analytes used in newly formed groups that are retained in final table
' (blue, "NumChemSumRetain"),
' (3) newly formed analyte groups (green, "NumNewGroups"),
' (4) analytes used in newly formed groups that are omitted from the final table
' (orange, "NumChemSumOmit"),
' (5) analytes that were omitted by choice (dark gray, "NumChemDeleted"), and
' (6) analytes that have too few data points (slash marks over any of the colors
' in groups 1 - 4 above, "NumChemTooFewData").
```

```
RawDataTableSort.Sort _
    Key1:=Range("NumColumnSort"), Order1:=xlAscending, _
    Key2:=Range("SummaryHeader"), Order2:=xlAscending, _
    Orientation:=xlLeftToRight
```

NOTE: This is the only place in the FPMData program where I used cell number references (C20, D20, and E20) instead of named ranges to identify a location in the spreadsheet. These are the locations of the first data cells in the Survey, Station, and Sample columns of the "Full Data Set Sorted by Analyte" table on the [AnalyteSummary] worksheet. Therefore, if changes are ever made to the [AnalyteSummary] worksheet, such as adding or deleting columns or rows, you might have to change the cells listed after Key1, Key2, and Key3.

```
FullRawDataTableNoHeader.Sort _
    Key1:=Range("C20"), Order1:=xlAscending, _
    Key2:=Range("D20"), Order2:=xlAscending, _
    Key3:=Range("E20"), Order3:=xlAscending, _
    Header:=xlNo, _
    Orientation:=xlTopToBottom
```

```
' Count the number of data points screened out for having unacceptable data quality.
```

```
NumDataBadDQs = Application.CountIf(ScreenedOut2, "DQ")
```

```
End Sub
```

This is the end of FPMData.xls mod03SummarizeData.

3.2.5 mod04MakeDataTable

This module contains only 1 macro, **MakeDataTable**. The code and comments are shown below.

```
Option Private Module
Option Explicit

' *****
' MakeDataTable() copies the information from [AnalyteSummary] to [DataTable]
' leaving out data points that either do not meet the specified data
' qualifiers and analytes that are being omitted.
' *****

Sub MakeDataTable()

' NetNumChemInitial = unsummed analytes with sufficient data to be retained,
' NumChemSumRetained = summed analytes retained by choice,
' NumNewGroupsRetained = new groups with sufficient data to be retained,
' NumChemSumOmitted = summed analytes omitted by default,
' NumChemOmitted = analytes omitted by choice,
' NumChemTooFewData = analytes omitted for having too few data, and
' NumNewGroupsOmitted = new groups omitted for having too few data.

NetNumChemInitial = Application.CountIf(NumColumnSort, 1)
NumChemSumRetained = Application.CountIf(NumColumnSort, 2)
NumNewGroupsRetained = Application.CountIf(NumColumnSort, 4)
NumChemSumOmitted = Application.CountIf(NumColumnSort, 8)
NumChemSumOmitted = NumChemSumOmitted + Application.CountIf(NumColumnSort, 40)
NumChemOmitted = Application.CountIf(NumColumnSort, 16)
NumChemTooFewData = Application.CountIf(NumColumnSort, 33)
NumChemTooFewData = NumChemTooFewData + Application.CountIf(NumColumnSort, 34)
NumNewGroupsOmitted = Application.CountIf(NumColumnSort, 36)
```

In the MakeDataTable Module, NumColumnSort values were assigned for each of the 6 following conditions:

Analyte put into a group but intentionally retained by user: NumColumnSort = 2
Analyte put into a group and not retained (default): NumColumnSort = 8
Analyte omitted on purpose by user: NumColumnSort = 16
Original analyte not used in a new group or omitted on purpose: NumColumnSort = 1
Newly formed group defined by the user: NumColumnSort = 4
Analyte with insufficient data: NumColumnSort = 32

All analytes started with a value of zero. When an analyte was found to meet one of the listed conditions the value of that condition was added to the relevant cell in the hidden NumColumnSort row. Therefore, analytes that fit more than one condition will have some combination of the 6 values given above. For example, an original analyte that was not used in a group or omitted is assigned a 1. If that analyte is later found to have too few data it is also assigned a 32. So, it ends up with NumColumnSort = 33. A newly formed group with insufficient data = 36, etc.

```
' Copy the ID information (survey, station, and sample numbers) from the
' [AnalyteSummary] table and paste it into the relevant columns in the [DataTable].
```

```
RawIDTable.Copy
IDTable.PasteSpecial Paste:=xlPasteValues
```

```
' Copy analyte names from [AnalyteSummary] to [DataTable] but only for the remaining
' original unscreened analytes (NetNumChemInitial), newly defined groups that have
' sufficient data (NumNewGroupsRetained), and analytes used in newly defined groups
' that the user retains on purpose (NumChemSumRetained). Check each data point in
' [AnalyteSummary] table to see if it should be omitted from the final data table due
' to poor DQ. If so, leave that cell in the [DataTable] blank; if not, copy it to
' [DataTable].
```



```

For i = 1 To NetNumChemInitial + NumNewGroupsRetained + NumChemSumRetained

    DataHeader(i) = RawDataHeader(i)
    DataUnits(i) = RawDataUnits(i)

    ' While transferring data, omit those that do not have acceptable DQs.
    ' (They had previously been marked with red cells.)

    For j = 1 To NumSamples
        If RawDataTable(j, i).Interior.ColorIndex = 3 Then
            DataTable(j, i) = ""
        Else
            DataTable(j, i) = RawDataTable(j, i)
        End If
    Next j

Next i

' Count the number of data points in each of the groups. Some of the groups
' may already have values at this stage, but this is after all of the data
' manipulations have been completed and should represent the final values.

' These columns hold the original analytes that were not screened or summed.
' Light yellow header (Sort Number = 1)

NumCol1 = 1
NumCol2 = Application.CountIf(NumColumnSort, 1)

If NumCol2 >= NumCol1 Then
    Range(NumFinalDataPoints(NumCol1), NumFinalDataPoints(NumCol2)).Select
    Set Workrange = Selection
    NetNumDataInitial = Application.Sum(Workrange)
Else: NetNumDataInitial = 0
End If

' These columns hold the newly defined groups.
' Blue header (Sort Number = 2)

NumCol1 = NumCol2 + 1
NumCol2 = NumCol2 + Application.CountIf(NumColumnSort, 2)

If NumCol2 >= NumCol1 Then
    Range(NumFinalDataPoints(NumCol1), NumFinalDataPoints(NumCol2)).Select
    Set Workrange = Selection
    NumDataSumRetained = Application.Sum(Workrange)
Else: NumDataSumRetained = 0
End If

' These columns hold the original analytes that were summed and retained.
' Green header (Sort Number = 4)

NumCol1 = NumCol2 + 1
NumCol2 = NumCol2 + Application.CountIf(NumColumnSort, 4)

If NumCol2 >= NumCol1 Then
    Range(NumFinalDataPoints(NumCol1), NumFinalDataPoints(NumCol2)).Select
    Set Workrange = Selection
    NumDataNewGroupsRetained = Application.Sum(Workrange)
Else: NumDataNewGroupsRetained = 0
End If

```

```

'   These columns hold the original analytes that were summed and omitted.
'   Orange header (Sort Number = 8)

NumCol1 = NumCol2 + 1
NumCol2 = NumCol2 + Application.CountIf(NumColumnSort, 8)

If NumCol2 >= NumCol1 Then
    Range(NumFinalDataPoints(NumCol1), NumFinalDataPoints(NumCol2)).Select
    Set Workrange = Selection
    NumDataSumOmitted = Application.Sum(Workrange)
Else: NumDataSumOmitted = 0
End If

'   These columns hold the original analytes that were deleted by choice.
'   Dark gray header (Sort Number = 16)

NumCol1 = NumCol2 + 1
NumCol2 = NumCol2 + Application.CountIf(NumColumnSort, 16)

If NumCol2 >= NumCol1 Then
    Range(NumFinalDataPoints(NumCol1), NumFinalDataPoints(NumCol2)).Select
    Set Workrange = Selection
    NumDataOmitted = Application.Sum(Workrange)
Else: NumDataOmitted = 0
End If

'   The following 4 column groups hold analytes omitted for having too few data.
'   Diagonal lines over any of the above colors (Sort Number = Original Sort Number + 32)

NumCol1 = NumCol2 + 1
NumCol2 = NumCol2 + Application.CountIf(NumColumnSort, 32)

If NumCol2 >= NumCol1 Then
    Range(NumFinalDataPoints(NumCol1), NumFinalDataPoints(NumCol2)).Select
    Set Workrange = Selection
    NumDataTooFewData = Application.Sum(Workrange)
Else: NumDataTooFewData = 0
End If

NumCol1 = NumCol2 + 1
NumCol2 = NumCol2 + Application.CountIf(NumColumnSort, 34)

If NumCol2 >= NumCol1 Then
    Range(NumFinalDataPoints(NumCol1), NumFinalDataPoints(NumCol2)).Select
    Set Workrange = Selection
    NumDataSumOmitted = NumDataSumOmitted + Application.Sum(Workrange)
End If

NumCol1 = NumCol2 + 1
NumCol2 = NumCol2 + Application.CountIf(NumColumnSort, 36)

If NumCol2 >= NumCol1 Then
    Range(NumFinalDataPoints(NumCol1), NumFinalDataPoints(NumCol2)).Select
    Set Workrange = Selection
    NumDataNewGroupsOmitted = Application.Sum(Workrange)
Else: NumDataNewGroupsOmitted = 0
End If

NumDataNewGroups = NumDataNewGroupsRetained + NumDataNewGroupsOmitted

NumCol1 = NumCol2 + 1
NumCol2 = NumCol2 + Application.CountIf(NumColumnSort, 40)

```

```

If NumCol2 >= NumCol1 Then
    Range(NumFinalDataPoints(NumCol1), NumFinalDataPoints(NumCol2)).Select
    Set Workrange = Selection
    NumDataSumOmitted = NumDataSumOmitted + Application.Sum(Workrange)
End If

'    Sort the columns in the final data table in order of analyte or group name.

Sheets("DataTable").Select

FinalDataTable.Sort _
    Key1:=DataHeader, Order1:=xlAscending, _
    Header:=xlGuess, _
    Orientation:=xlLeftToRight

```

After all of the sorting and counting has been completed, the code below checks to make sure that the sum of all of the groups equals the total. You can also do this by looking at the "Description of Data Set" in the [DataSetSummary] page. If everything balances, the numbers listed in the two entries named "Total Analytes and Groups" should be the same. The numbers listed in the two entries named "Total No. of Available Data" should also be the same.

```

'    Check to make sure that the number of initial groups plus the newly created groups
'    equals the final number of groups.  If not, give error message.

If NumChemInitial + NumNewGroups <> NumChemTooFewData + NumNewGroupsOmitted + _
    NumChemSumOmitted + NumChemOmitted + NetNumChemInitial + NumChemSumRetained + _
    NumNewGroupsRetained Then

    Sheets("DataSetSummary").Select: DataSummary.Select
    Comment = "'Total Analytes and Groups' do not balance.  Have all duplicates been
removed?"
    Call ShowMsg

End If

'    Check to make sure that the number of initial analytes plus the newly summed analytes
'    equals the final number of analytes.  If not, give error message.

If NumDataInitial + NumDataNewGroups <> NetNumDataInitial + NumDataSumRetained + _
    NumDataNewGroupsRetained + NumDataTooFewData + NumDataSumOmitted + NumDataOmitted + _
    NumDataBadDQs + NumDataNewGroupsOmitted Then

    Sheets("DataSetSummary").Select: DataSummary.Select
    Comment = "'Total No. of Available Data' do not balance.  Have all duplicates been
removed?"
    Call ShowMsg

End If

End Sub

```

This is the end of FPMDData.xls mod04MakeDataTable.

3.2.6 mod05MakeLists

This module contains 6 macros that are used to create some of the lists of analyte and data qualifier names from the raw data set. Some of the lists include check boxes, which allow you to select specific analytes or DQs that you would like to omit from the final data table.

GenerateLists calls the list-making routines from a button on the [OmitAnalytes] worksheet.

ChemList checks the data on the [InputData] page and generates a list of analytes on the [OmitAnalytes] worksheet.

DataQualList checks the data on the [InputData] page and generates a list of data qualifiers on the [OmitAnalytes] worksheet.

SumsChemList generates and formats a list of analytes on the [CreateSums] worksheet. This list is used to select the members of newly-defined groups.

GroupList generates a list of all newly-defined groups and the analytes that make up each group. This list is on the [DataSetSummary] page.

OmitList generates a list of all analytes and groups that have been screened out. This list is on the [DataSetSummary] page.

The code for these macros is shown below.

```
Option Private Module
Option Explicit
```

```
' The macros in this module are used to generate the lists of
' analytes and data qualifiers on the [OmitAnalytes] page as
' well as the analyte list on the [CreateSums] page.

' *****
' GenerateLists() calls the list-making routines from a button
' on the [OmitAnalytes] worksheet.
' *****

Sub GenerateLists()

    Call ChemList
    Call DataQualList

' After making lists, always leave [InputData] sorted in ascending
' order by Survey, Station, and Sample.

FullInputTable.Sort _
    Key1:=InputSurvey, Order1:=xlAscending, _
    Key2:=InputStation, Order2:=xlAscending, _
    Key3:=InputSample, Order3:=xlAscending, _
    Header:=xlYes, _
    Orientation:=xlTopToBottom

End Sub
```

```

' *****
' ChemList() checks the data on the [InputData] page and
' generates a list of analytes on the [OmitAnalytes] worksheet.
' *****

Sub ChemList()

Application.ScreenUpdating = False
Call SetRanges

' Clear the existing analytes list, ...

NameScreenedAnalytes.ClearContents
AnalyteNameList.ClearContents
AnalyteNameUnits.ClearContents

' ... uncheck all boxes, and ...

AnalyteOmitState = "#N/A"
AnalyteRetainState = "#N/A"

' ... check the current capacity of the analytes list table.

ListCapacity = Application.CountBlank(AnalyteNameList)

' Sort the InputData by analyte name.

FullInputTable.Sort _
    Key1:=InputName, Order1:=xlAscending, _
    Header:=xlYes, _
    Orientation:=xlTopToBottom

' Scan the analyte names in InputData. Whenever a new name
' is encountered, count it and copy that name to the Analytes List.

NumChemInitial = 0
For i = 1 To Application.CountA(InputName)
    If InputName(i) <> InputName(i - 1) Then
        NumChemInitial = NumChemInitial + 1

' Stop if the number of analytes exceeds the capacity of the Analytes List.

        If NumChemInitial > ListCapacity Then
            MsgBox ("Number of analytes exceeds length of 'Analytes List'")
            AnalyteNameList.Select
            Application.StatusBar = "": End
        End If

        AnalyteNameList(NumChemInitial) = InputName(i)
        AnalyteNameUnits(NumChemInitial) = InputUnits(i)
        AnalyteOmitState(NumChemInitial) = "False"
        AnalyteRetainState(NumChemInitial) = "False"

    End If
Next i

' Move the cursor off the screen and turn on screen updating

Range("A1").Select
Application.ScreenUpdating = True

End Sub

```

```

' *****
' DataQualList() checks the data on the [InputData] page and generates
' a list of data qualifiers on the [OmitAnalytes] worksheet.
' *****

Sub DataQualList()

Application.ScreenUpdating = False
Call SetRanges

' Clear the existing list, uncheck all boxes, and
' check the current capacity of the DQ list table.

QualList.ClearContents
QualState = "#N/A"
ListCapacity = Application.CountBlank(QualList)

' Sort the InputData table by analyte name.

FullInputTable.Sort _
    Key1:=InputQual, Order1:=xlAscending, _
    Header:=xlYes, _
    Orientation:=xlTopToBottom

' Scan the data quality symbols in InputData. Whenever a new one
' is encountered, count it and copy that name to the Analytes List.

NumQual = 0
For i = 1 To Application.CountA(InputQual)

    If InputQual(i) <> InputQual(i - 1) Then
        NumQual = NumQual + 1

' Stop if the number of DQs exceeds the capacity of the Data Qualifiers List.

        If NumQual > ListCapacity Then
            MsgBox ("Number of Data Qualifiers exceeds length of 'Data Qualifiers List'")
            QualList.Select
            Application.StatusBar = "": End
        End If

        QualList(NumQual) = InputQual(i)
        QualState(NumQual) = "False"

    End If

Next i

' Move the cursor off the screen and turn on screen updating

Range("A1").Select
Application.ScreenUpdating = True

End Sub

```



```

' *****
' SumsChemList() generates and formats a list of analytes
' on the [CreateSums] worksheet. This list is used to select
' the members of newly-defined groups.
' *****

Sub SumsChemList()

    Dim NumMemberList As Integer

    Application.ScreenUpdating = False

    Call SetRanges
    NumChemInitial = Application.CountA(AnalyteNameList)

' Delete all entries from the first three columns of the Analyte List
' table, color the cells light gray, and set the font color to black.

    SumListNums.ClearContents
    SumListNums.Interior.ColorIndex = 15
    SumListNums.Font.ColorIndex = 1
    SumListNames.ClearContents
    SumListNames.Interior.ColorIndex = 15
    SumListNames.Font.ColorIndex = 1
    SumListUnits.ClearContents
    SumListUnits.Interior.ColorIndex = 15
    SumListUnits.Font.ColorIndex = 1
    SumGroupNames.ClearContents
    GroupNameUnits.ClearContents

' Put the name of each analyte into the table. If it has been
' selected for omission, color the cells dark gray and set
' the font color to white. If not, color the cells light yellow
' and set the font color to black.

    For i = 1 To NumChemInitial
        SumListNames(i) = AnalyteNameList(i)
        SumListUnits(i) = AnalyteNameUnits(i)
        AnalyteRetainState(i) = "False"
        If AnalyteOmitState(i) = "True" Then
            SumListNames(i).Interior.ColorIndex = 16
            SumListNames(i).Font.ColorIndex = 2
            SumListNums(i).Interior.ColorIndex = 16
            SumListNums(i).Font.ColorIndex = 2
            SumListUnits(i).Interior.ColorIndex = 16
            SumListUnits(i).Font.ColorIndex = 2
        Else
            SumListNames(i).Interior.ColorIndex = 36
            SumListNames(i).Font.ColorIndex = 1
            SumListNums(i).Interior.ColorIndex = 36
            SumListNums(i).Font.ColorIndex = 1
            SumListUnits(i).Interior.ColorIndex = 36
            SumListUnits(i).Font.ColorIndex = 1
        End If
    Next i

```

```

If RetainGroupList = "True" Then

    NumMemberList = Application.CountA(GroupMemberList)

    m = 0
    For i = 1 To NumMemberList
        If GroupMemberList(i).Interior.ColorIndex = 36 Then
            m = m + 1
            SumGroupNames(m) = GroupMemberList(i)
        Else
            For n = 1 To NumChemInitial
                If SumListNames(n) = GroupMemberList(i) Then
                    SumListNums(n) = m
                    If GroupMemberRetain(i) = "True" Then AnalyteRetainState(n) = "True"
                    GroupNameUnits(m) = SumListUnits(i)
                    Exit For
                End If
            Next n
        End If
    Next i

End If

Range("A1").Select

End Sub

' *****
' GroupList() generates a list of all newly-defined groups
' and the analytes that make up each group. This list is
' on the [DataSetSummary] page.
' *****

Sub GroupList()

Dim Count As Integer

' Delete all entries from the Members of Defined Groups list,
' make the empty cells light gray, and make sure none of the
' fonts are set to bold.

Count = 0
Sheets("DataSetSummary").Activate
GroupMemberList.Select
    With Selection
        .ClearContents
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlBottom
        .Interior.ColorIndex = 15
        .Font.Bold = False
    End With
GroupMemberRetain.ClearContents

For i = 1 To NumNewGroups

```

```

' Copy one of the new group names into the list, make that
' cell light yellow and make the font bold.

Count = Count + 1
GroupMemberList(Count) = SumGroupNames(i)
GroupMemberList(Count).Interior.ColorIndex = 36
GroupMemberList(Count).Font.Bold = True

' Copy the names of all the members of the previously listed
' group into the list below the group name.

For j = 1 To NumChemInitial
    If SumListNums(j) = i Then
        Count = Count + 1
        GroupMemberList(Count) = SumListNames(j)
        If AnalyteRetainState(j) = "True" Then GroupMemberRetain(Count) = "True"
    End If
Next j

Next i

Range("A1").Select

End Sub

' *****
' OmitList() generates a list of all analytes and groups that have
' been screened out. This list is on the [DataSetSummary] page.
' *****

Sub OmitList()

Dim OmitFirst As Integer, OmitLast As Integer

' The analytes in the Data Table are listed such that all of the saved
' ones come first (i.e, NetNumChemInitial, NumChemSumRetained, and
' NumNewGroupsRetained) followed by the omitted ones. Therefore, the
' first omitted analyte is the number of saved analytes + 1.

OmitFirst = NetNumChemInitial + NumChemSumRetained + NumNewGroupsRetained + 1
OmitLast = NumChemInitial + NumNewGroups

j = 1
For i = OmitFirst To OmitLast
    NameScreenedAnalytes(j) = RawDataHeader(i)
    j = j + 1
Next i

End Sub

```

This is the end of FPMDData.xls mod05MakeLists.

3.2.7 mod06FormatData

This module contains 16 macros, most of which are used to delete data from tables or format data in tables. Many of these macros are simply to make the final tables easier to read.

Auto_Open sets up the appearance of the workbook upon opening.

SetUpScreen modifies the default screen appearance. It's called from the Auto_Open procedure.

StripBlanks removes leading and trailing blanks from the text entries in the Initial Analytical Data table. In trial runs, some data were found to contain unexpected blanks, which caused the program to treat some equivalent entries as different entries (i.e., "SampleName " appears to differ from "SampleName").

CenterText centers selected text and adjusts columns to appropriate widths.

ClearAllTables deletes entries from all tables except [InputData]. It's called from a button on the [OmitAnalytes] page.

ClearRawAndDataTables deletes all entries from the [AnalyteSummary] and [DataTable] worksheets and restores cell formats where necessary.

ClearSumLists deletes entries from tables on the [CreateSums] worksheet, restores the default color (gray) to the cells, and resets the font color to black. It's called from a button on [CreateSums].

ClearInitialAnalyticalData deletes all of the entries from the Initial Analytical Data Table on the [InputData] worksheet. It's called from a button on the [InputData] worksheet.

ClearScreenedCodes deletes only the entries from the three columns of screened data codes on [InputData]. It's called from a button on the [InputData] worksheet.

DataFormat formats the data tables so that the columns are an appropriate width to make them easier to read.

SortInputData sorts the data in the Input Table in order of survey, station, sample, and analyte.

ColorKey shows a userform that explains the colors used in cells on [AnalyteSummary] page.

CodeKey shows a userform that explains the codes in the Screened Data Codes table.

SaveGroup saves group and analyte name information for future use.

RetrieveGroup copies previously-saved group name and analyte information for reuse.

Auto_Close restores original toolbars, menus, etc. It works automatically on closing.

The code for these macros is shown below.

```
Option Private Module
Option Explicit
```

```
' The macros in this module are used to format the worksheets and
' perform basic tasks like deleting all entries in a specified table.
```

```

' *****
' Auto_Open() sets up the appearance of the workbook upon opening.
' *****

Sub Auto_Open()

Application.ScreenUpdating = False
Application.StatusBar = ""

Call SetRanges
Call SetUpScreen

' This sets each worksheet back to a standard top-left
' orientation with the cursor hidden off screen and hides
' the worksheet that stores reference information [Names].

For i = 1 To 6
    Sheets(i).Activate
    Application.DisplayCommentIndicator = xlCommentIndicatorOnly
    ActiveWindow.ScrollColumn = 1
    ActiveWindow.ScrollRow = 1
    Range("A1").Select
Next i

Sheets(7).Visible = False

Sheets("InputData").Activate
Application.ScreenUpdating = True

End Sub

' *****
' SetUpScreen() modifies the default screen appearance.
' It's called from the Auto_Open procedure.
' *****

Sub SetUpScreen()

Dim Bar As Object, BarCount As Integer

    For i = 1 To 7
        Sheets(i).Activate
        ActiveWindow.DisplayHeadings = False
    Next i

' To create more space, hide the Formula Bar and the Comment Indicator

Application.DisplayCommentIndicator = False
Application.DisplayFormulaBar = False

' Keep track of what toolbars are open so they can be reopened
' before exiting. Names are stored on the hidden worksheet [Names].
' Then hide the open toolbars to create more space on the screen.

ToolBarStorage.ClearContents
BarCount = 0

```

```

For Each Bar In Toolbars
    If Bar.Visible = True Then
        BarCount = BarCount + 1
        ToolBarStorage(BarCount) = Bar.Name
        Bar.Visible = False
    End If
Next Bar

End Sub

' *****
' StripBlanks() removes leading and trailing blanks from the text
' entries in the Initial Analytical Data table. In trial runs,
' some data were found to contain unexpected blanks, which caused
' the program to treat some equivalent entries as different
' entries (i.e., "SampleName " appears to differ from "SampleName").
' *****

Sub StripBlanks()

Application.ScreenUpdating = False
Call SetRanges
NumDataInitial = Application.CountA(InputName)
Application.StatusBar = "Removing Leading and Trailing Blanks from Data Set"

For i = 1 To NumDataInitial
    InputSurvey(i) = Trim(InputSurvey(i))
    InputStation(i) = Trim(InputStation(i))
    InputSample(i) = Trim(InputSample(i))
    InputName(i) = Trim(InputName(i))
    InputUnits(i) = Trim(InputUnits(i))
    InputQual(i) = Trim(InputQual(i))
Next i

Application.ScreenUpdating = True
Application.StatusBar = False
MsgBox ("    Blanks Have Been Removed    ")

End Sub

' *****
' CenterText() centers selected text and adjusts columns
' to appropriate widths.
' *****

Sub CenterText()

With Selection
    .HorizontalAlignment = xlCenter
    .Columns.AutoFit
End With

Range("A1").Select

End Sub

```

```

' *****
' ClearAllTables() deletes entries from all tables except
' [InputData]. Called from a button on [OmitAnalytes].
' *****

Sub ClearAllTables()

Call SetRanges

Application.ScreenUpdating = False

' Delete entries from Analyte List and Qualifier List on
' [OmitAnalytes] and restore all checkboxes to n/a
' (neither checked or unchecked). Also delete entries in
' the ScreenedOut table and Duplicate Samples table on
' the [InputData] page and the lists on [DataSetSummary] page.

AnalyteNameList.ClearContents
AnalyteNameUnits.ClearContents
AnalyteOmitState = "#N/A"
QualList.ClearContents
QualState = "#N/A"
NameScreenedAnalytes.ClearContents
DataSummary.ClearContents
ScreenedOut.ClearContents
DupeTable.ClearContents

If RetainGroupList = "False" Then
    GroupMemberList.ClearContents
    GroupMemberList.Interior.ColorIndex = 15
End If

Call ClearRawAndDataTables
Call ClearSumLists

Range("A1").Select
Application.ScreenUpdating = True

End Sub

' *****
' ClearRawAndDataTables() deletes all entries from the
' [AnalyteSummary] and [DataTable] worksheets and restores
' cell formats where necessary.
' *****

Sub ClearRawAndDataTables()

' Delete data and restore the format of Raw Data Table

RawDataTable.ClearContents
RawDataTable.Interior.ColorIndex = xlNone
RawDataHeader.ClearContents
RawDataHeader.Interior.ColorIndex = 15
RawDataHeader.Font.ColorIndex = 1
RawDataHeader.Interior.Pattern = xlSolid
RawDataUnits.ClearContents
RawIDTable.ClearContents
NumColumnSort.ClearContents

```

```

' Delete data and restore the format of Summary Table

FullSummaryTable.ClearContents
SummaryHeader.Interior.ColorIndex = 15
SummaryHeader.Font.ColorIndex = 1
SummaryHeader.Interior.Pattern = xlSolid

' Delete data and restore the format of Data Table

DataHeader.ClearContents
DataTable.ClearContents
DataUnits.ClearContents
IDTable.ClearContents

End Sub

' *****
' ClearSumLists() deletes entries from tables on the [CreateSums]
' worksheet, restores the default color (gray) to the cells, and
' resets the font color to black. Called from button on [CreateSums].
' *****

Sub ClearSumLists()

Application.ScreenUpdating = False

Call SetRanges

SumGroupNames.ClearContents
SumGroupNames.Interior.ColorIndex = 15
SumGroupNames.Font.ColorIndex = 1
SumListNames.ClearContents
SumListNames.Interior.ColorIndex = 15
SumListNames.Font.ColorIndex = 1
SumListNums.ClearContents
SumListNums.Interior.ColorIndex = 15
SumListNums.Font.ColorIndex = 1
SumListUnits.ClearContents
SumListUnits.Interior.ColorIndex = 15
SumListUnits.Font.ColorIndex = 1
AnalyteRetainState = "#N/A"
GroupNameUnits.ClearContents

Range("A1").Select
Application.ScreenUpdating = True

End Sub

```



```

' *****
' ClearInitialAnalyticalData() deletes all of the entries from
' the Initial Analytical Data Table on the [InputData] worksheet.
' It's called from the button on the [InputData] worksheet.
' *****

Sub ClearInitialAnalyticalData()

Dim InputTable As Range
Dim Msg As String, Title As String
Dim Response As Integer

Set InputTable = Range("InputTable")

' Make sure the user really wants to delete all of the input data

Msg = "Are you sure you want to delete ALL of the data from the Initial Analytical Data
Table?"
Title = "FPM Data Module: Input Data Screen"
Response = MsgBox(Msg, vbYesNo + vbDefaultButton2, Title)

If Response = vbYes Then InputTable.ClearContents

End Sub

' *****
' ClearScreenedCodes() deletes only the entries from the
' three columns of screened data codes on [InputData].
' It's called from the button on the [InputData] worksheet.
' *****

Sub ClearScreenedCodes()

Range("ScreenedOut").ClearContents

End Sub

' *****
' DataFormat() formats the data tables so that the columns
' are an appropriate width to make them easier to read.
' *****

Sub DataFormat()

' Leave [InputData] sorted in ascending order by
' Survey, Station, and Sample.

FullInputTable.Sort _
    Key1:=InputSurvey, Order1:=xlAscending, _
    Key2:=InputStation, Order2:=xlAscending, _
    Key3:=InputSample, Order3:=xlAscending, _
    Header:=xlYes, _
    Orientation:=xlTopToBottom

```

```

'   Center the text in the [InputTable] worksheet.

Sheets("InputData").Select
    Range("InputHeader", "InputTable").Select
    Call CenterText

'   Center the text in the [AnalyteSummary] worksheet.

Sheets("AnalyteSummary").Select

    FullRawDataTable.Select
    Call CenterText

    RawIDTable.ColumnWidth = 12

    For i = 1 To NumChemInitial + NumNewGroups
        SummaryHeader(i).Select
        With Selection
            If .ColumnWidth < 10 Then .ColumnWidth = 10
        End With
    Next i

Sheets("DataTable").Select

    FullDataTable.Select
    Call CenterText

    IDTable.ColumnWidth = 12

    For i = 1 To NetNumChemInitial + NumChemSumRetained + NumNewGroups
        DataHeader(i).Select
        With Selection
            If .ColumnWidth < 10 Then .ColumnWidth = 10
        End With
    Next i

For i = 5 To 1 Step -1
    Sheets(i).Select
    Range("A1").Select
Next i

End Sub

'   *****
'   SortInputData() sorts the data in the Input Table
'   in order of survey, station, sample, and analyte.
'   *****

Sub SortInputData()

'   The sort function can only handle three sort factors at
'   a time, so first sort by the fourth one (analyte name)
'   and then sort by the first three (survey, station, sample).

FullInputTable.Sort _
    Key1:=InputName, Order1:=xlAscending, _
    Header:=xlYes, _
    Orientation:=xlTopToBottom

```

```

FullInputTable.Sort _
    Key1:=InputSurvey, Order1:=xlAscending, _
    Key2:=InputStation, Order2:=xlAscending, _
    Key3:=InputSample, Order3:=xlAscending, _
    Header:=xlYes, _
    Orientation:=xlTopToBottom

End Sub

' *****
' ColorKey() shows a userform that explains
' the colors used in cells on [AnalyteSummary] page.
' *****

Sub ColorKey()
    CellColors.Show
End Sub

' *****
' CodeKey() shows a userform that explains
' the codes in the Screened Data Codes table.
' *****

Sub CodeKey()
    DataCodes.Show
End Sub

' *****
' SaveGroup() saves group and analyte name
' information for future use.
' *****

Sub SaveGroup()

    Dim ProgramName As String
    Dim DestinationGroupName As String, DestinationFileName As String
    Dim DestinationGroup As String, DestinationRetain As String
    Dim SaveGroupName As Range, SaveFileName As Range
    Dim CurrentGroups As Range
    Dim i As Integer, ListLength As Integer

    Set SaveGroupName = Range("SaveGroupName")
    Set SaveFileName = Range("SaveFileName")

    Range("A1").Select
    Application.ScreenUpdating = False

    Call SetRanges

    ProgramName = ActiveWorkbook.Name
    DestinationFileName = SaveFileName
    DestinationGroupName = SaveGroupName

    On Error GoTo EndThis

    GroupMemberList.Copy

    Windows(DestinationFileName).Activate
    Sheets("FPMDDataGroups").Select

```

```

Set CurrentGroups = Range("CurrentGroups")
ListLength = Application.CountA(CurrentGroups) + Application.CountBlank(CurrentGroups)

For i = 1 To ListLength
    If CurrentGroups(i) = "" Then Exit For
    If i = ListLength Then
        MsgBox ("There is no available storage space in the destination file.")
        End
    End If
Next i

DestinationGroupName = "Name" & i
Range(DestinationGroupName) = SaveGroupName
CurrentGroups(i) = SaveGroupName

DestinationGroup = "Group" & i
Range(DestinationGroup).Select
Selection.PasteSpecial Paste:=xlPasteAllExceptBorders, Operation:=xlNone, _
    SkipBlanks:=False, Transpose:=False
Application.CutCopyMode = False

Windows(ProgramName).Activate
GroupMemberRetain.Copy

Windows(DestinationFileName).Activate
Sheets("FPMDDataGroups").Select

DestinationRetain = "Retain" & i
Range(DestinationRetain).Select
Selection.PasteSpecial Paste:=xlPasteAllExceptBorders, Operation:=xlNone, _
    SkipBlanks:=False, Transpose:=False
Application.CutCopyMode = False

Range("A1").Select

Windows(ProgramName).Activate

Application.ScreenUpdating = True

Exit Sub

EndThis:

Application.CutCopyMode = False
Range("A1").Select

MsgBox _
("An error was encountered in this procedure." & Chr(13) & _
"    Make sure that the target file is open" & Chr(13) & _
"    and in the same folder as this spreadsheet," & Chr(13) & _
"    and/or check the spelling and try again.")

End Sub

```

```

' *****
' RetrieveGroup() copies previously-saved
' group name and analyte information for reuse.
' *****

Sub RetrieveGroup()

Dim ProgramName As String
Dim SourceGroupName As String, SourceFileName As String
Dim SourceGroup As String, SourceRetain As String
Dim RetrieveGroupName As Range, RetrieveFileName As Range
Dim CurrentGroups As Range
Dim i As Integer, ListLength As Integer

Set RetrieveGroupName = Range("RetrieveGroupName")
Set RetrieveFileName = Range("RetrieveFileName")

Range("A1").Select
Application.ScreenUpdating = False

Call SetRanges
If AnalyteNameList(1) = "" Then
    Sheets("OmitAnalytes").Select
    Application.ScreenUpdating = True
    MsgBox ("You must generate data lists on this page before you can proceed.")
    End
End If

ProgramName = ActiveWorkbook.Name
SourceFileName = RetrieveFileName
SourceGroupName = RetrieveGroupName

On Error GoTo EndThis

Windows(SourceFileName).Activate
Sheets("FPMDDataGroups").Select
Set CurrentGroups = Range("CurrentGroups")
ListLength = Application.CountA(CurrentGroups) + Application.CountBlank(CurrentGroups)

For i = 1 To ListLength
    If CurrentGroups(i) = SourceGroupName Then Exit For
    If i = ListLength Then
        MsgBox ("The designated group was not found in the source file.")
        Windows(ProgramName).Activate
        End
    End If
Next i

SourceGroup = "Group" & i
Range(SourceGroup).Copy

Windows(ProgramName).Activate
GroupMemberList.Select
Selection.PasteSpecial Paste:=xlPasteAllExceptBorders, Operation:=xlNone, _
    SkipBlanks:=False, Transpose:=False
Application.CutCopyMode = False

Windows(SourceFileName).Activate
Sheets("FPMDDataGroups").Select

SourceRetain = "Retain" & i
Range(SourceRetain).Copy

```

```

Windows (ProgramName).Activate
GroupMemberRetain.Select
Selection.PasteSpecial Paste:=xlPasteAllExceptBorders, Operation:=xlNone, _
    SkipBlanks:=False, Transpose:=False
Application.CutCopyMode = False

Sheets("CreateSums").Select
RetainGroupList = "True"
Call SumsChemList
RetainGroupList = "False"
Range("A1").Select

Application.ScreenUpdating = True
Exit Sub

EndThis:

Application.CutCopyMode = False
Range("A1").Select

MsgBox _
("An error was encountered in this procedure." & Chr(13) & _
"    Make sure that the source file is open" & Chr(13) & _
" and in the same folder as this spreadsheet," & Chr(13) & _
"    and/or check the spelling and try again.")

End Sub

' *****
' Auto_Close() restores original toolbars, menus, etc.
' Works automatically on closing.
' *****

Sub Auto_Close()

    Call SetRanges
    On Error Resume Next
    Application.ScreenUpdating = False
    Sheets("InputData").Select

    For i = 1 To Application.CountA(ToolBarStorage)
        Toolbars(ToolBarStorage(i).Value).Visible = True
    Next i

    Application.DisplayFormulaBar = True
    Application.DisplayCommentIndicator = True

End Sub

```

This is the end of FPMData.xls mod06FormatData.

This is the end of macro code for the FPMData.xls Worksheet.
--

3.3 Macro Code for FPMDataGroups.xls

3.3.1 Summary

FPMDataGroups.xls contains 3 Microsoft Excel® Visual Basic macros that perform specific functions for the spreadsheet. The macros are stored in one module. The code along with some explanatory text is listed in Section 3.4.2. No password is required to access the module in the spreadsheet.

3.3.1.1 Macros Activated When Spreadsheet Opens or Closes

MacroName() Module Containing Macro	Macro Action
Auto_Open() Module1	When you open the spreadsheet this macro (1) calls the macro SetRanges() in Module1 to initialize all of the variables, and (2) calls the macro Refresh() in Module1 to restore the appearance of the tables to their default settings.

3.3.1.2 Macros Activated from Buttons on Worksheets

MacroName() Module Containing Macro	Button that Activates Macro [Worksheet with Button]	Macro Action
Refresh() Module1	Refresh Worksheet [FPMDataGroups]	Deletes group data for any group that has had its name deleted from the table of "Groups Currently Stored in this Worksheet."

3.3.2 Module1

```
Option Explicit
Option Private Module
```

```
Dim CurrentGroups As Range
```

```
Dim Group1 As Range, Group2 As Range, Group3 As Range, Group4 As Range
Dim Group5 As Range, Group6 As Range, Group7 As Range, Group8 As Range
Dim Group9 As Range, Group10 As Range, Group11 As Range, Group12 As Range
Dim Group13 As Range, Group14 As Range, Group15 As Range, Group16 As Range
Dim Group17 As Range, Group18 As Range, Group19 As Range, Group20 As Range
```

```
Dim Name1 As Range, Name2 As Range, Name3 As Range, Name4 As Range
Dim Name5 As Range, Name6 As Range, Name7 As Range, Name8 As Range
Dim Name9 As Range, Name10 As Range, Name11 As Range, Name12 As Range
Dim Name13 As Range, Name14 As Range, Name15 As Range, Name16 As Range
Dim Name17 As Range, Name18 As Range, Name19 As Range, Name20 As Range
```

```
Dim Retain1 As Range, Retain2 As Range, Retain3 As Range, Retain4 As Range
Dim Retain5 As Range, Retain6 As Range, Retain7 As Range, Retain8 As Range
Dim Retain9 As Range, Retain10 As Range, Retain11 As Range, Retain12 As Range
Dim Retain13 As Range, Retain14 As Range, Retain15 As Range, Retain16 As Range
Dim Retain17 As Range, Retain18 As Range, Retain19 As Range, Retain20 As Range
```

```
Dim i As Integer, CurrentName As String, CurrentList As String
```

```
' *****
' Auto_Open() sets up the appearance of the workbook upon opening.
' *****
```

```
Sub Auto_Open()
```

```
Call Refresh
```

```
End Sub
```

```
' *****
' Refresh() deletes group data for any group that has had its
' name deleted from the table of group names.
' *****
```

```
Sub Refresh()
```

```
Call SetRanges
```

```
Application.ScreenUpdating = False
```

```
For i = 1 To 20
```

```
    If CurrentGroups(i) = "" Then
        CurrentName = "Name" & i
        CurrentList = "Group" & i
        Range(CurrentName).ClearContents
        Range(CurrentList).ClearContents
        Range(CurrentList).Select
        With Selection.Interior
            .ColorIndex = 15
            .Pattern = xlSolid
            .PatternColorIndex = xlAutomatic
        End With
    End If
```

```
Next i
```

```
Range("A1").Select
Application.ScreenUpdating = True
```

```
End Sub
```

```
' *****
' SetRanges() initializes the variables by setting them equal
' to the values in the previously-declared ranges.
' *****
```

```
Sub SetRanges()
```

```
Set CurrentGroups = Range("CurrentGroups")
```

```
Set Group1 = Range("Group1")
Set Group2 = Range("Group2")
Set Group3 = Range("Group3")
Set Group4 = Range("Group4")
Set Group5 = Range("Group5")
Set Group6 = Range("Group6")
Set Group7 = Range("Group7")
Set Group8 = Range("Group8")
Set Group9 = Range("Group9")
Set Group10 = Range("Group10")
```



```

Set Group11 = Range("Group11")
Set Group12 = Range("Group12")
Set Group13 = Range("Group13")
Set Group14 = Range("Group14")
Set Group15 = Range("Group15")
Set Group16 = Range("Group16")
Set Group17 = Range("Group17")
Set Group18 = Range("Group18")
Set Group19 = Range("Group19")
Set Group20 = Range("Group20")

```

```

Set Name1 = Range("Name1")
Set Name2 = Range("Name2")
Set Name3 = Range("Name3")
Set Name4 = Range("Name4")
Set Name5 = Range("Name5")
Set Name6 = Range("Name6")
Set Name7 = Range("Name7")
Set Name8 = Range("Name8")
Set Name9 = Range("Name9")
Set Name10 = Range("Name10")
Set Name11 = Range("Name11")
Set Name12 = Range("Name12")
Set Name13 = Range("Name13")
Set Name14 = Range("Name14")
Set Name15 = Range("Name15")
Set Name16 = Range("Name16")
Set Name17 = Range("Name17")
Set Name18 = Range("Name18")
Set Name19 = Range("Name19")
Set Name20 = Range("Name20")

```

```

Set Retain1 = Range("Retain1")
Set Retain2 = Range("Retain2")
Set Retain3 = Range("Retain3")
Set Retain4 = Range("Retain4")
Set Retain5 = Range("Retain5")
Set Retain6 = Range("Retain6")
Set Retain7 = Range("Retain7")
Set Retain8 = Range("Retain8")
Set Retain9 = Range("Retain9")
Set Retain10 = Range("Retain10")
Set Retain11 = Range("Retain11")
Set Retain12 = Range("Retain12")
Set Retain13 = Range("Retain13")
Set Retain14 = Range("Retain14")
Set Retain15 = Range("Retain15")
Set Retain16 = Range("Retain16")
Set Retain17 = Range("Retain17")
Set Retain18 = Range("Retain18")
Set Retain19 = Range("Retain19")
Set Retain20 = Range("Retain20")

```

```

End Sub

```

This is the end of macro code for the FPMDDataGroups.xls Worksheet.

3.4 Macro Code for FPMAnova.xls

3.4.1 Summary

FPMAnova.xls contains 21 Microsoft Excel® Visual Basic macros that perform specific functions for the spreadsheet. The macros are stored in four modules. The code from each module along with some explanatory text is listed in Sections 3.4.2 to 3.4.5. A password is required to access the modules in the spreadsheet.

3.4.1.1 Macros Activated When Spreadsheet Opens or Closes

MacroName() Module Containing Macro	Macro Action
Auto_Open() mod04FormatPages	When you open the spreadsheet this macro (1) calls the macro SetRanges() in mod01RunProgram to initialize all of the variables, (2) calls the macro SetUpScreen() in mod04FormatPages to close unneeded toolbars, and (3) resets all worksheets to the standard top-left orientation.
Auto_Close() mod04FormatPages	When you close the spreadsheet this macro restores the toolbars that were closed when the program opened and resets the formats of the worksheets.

3.4.1.2 Macros Activated from Buttons on Worksheets

MacroName() Module Containing Macro	Button that Activates Macro [Worksheet with Button]	Macro Action
RunProgram() mod01RunProgram	Run ANOVA [ControlScreen]	See Section 3.4.1.3
ClearAllTables() mod04FormatPages	Clear All Workbook Tables [ControlScreen]	Deletes entries from all tables. Calls ClearFinalTable() in mod04FormatPages to delete entries from the [FinalDataSet] page.
ClearChem() mod04FormatPages	Clear ChemData Table [ControlScreen]	Deletes entries from [ChemData] page.
ClearBio() mod04FormatPages	Clear BioData Table [ControlScreen]	Deletes entries from [BioData] page.
ClearSorted() mod04FormatPages	Clear SortedData Table [ControlScreen]	Deletes entries from [SortedData] page.
ClearAnova() mod04FormatPages	Clear AnovaResults Table [ControlScreen]	Deletes entries from [AnovaResults] page.
DeleteStoredData() mod01RunProgram	Clear Saved Sorted Data Tables [ControlScreen]	Deletes data from the Sorted Data Tables. Renames the worksheets with numbers 01, 02, etc., and hides the now-empty worksheets.
CopySortedData() mod01RunProgram	Copy Screened Data [ControlScreen]	Copies data from [DataTable] page of the FPMData.xls spreadsheet and pastes it into the [ChemData] page of the FPMAnova.xls spreadsheet

MacroName() Module Containing Macro	Button that Activates Macro [Worksheet with Button]	Macro Action
CompileDataSet() mod01RunProgram	Compile Data Set [AnovaResults]	After you select the bio data set you want to use based on the Anova test results, this compiles a final data table made up of the analytical data selected in FPMData.xls and the bio data selected in FPMAnova.xls. The data are stored on the [FinalDataSet] page.

3.4.1.3 Macros that Control the Main Function of the Program

When you click on the “**Run ANOVA**” button, which is located on the [ControlScreen] worksheet, you start the RunProgram() macro in the mod01RunProgram module. This macro performs some basic functions on its own and also calls other macros that each carry out part of the overall process. In the table below the RunProgram() macro code is shown in the left column and the steps that are carried out are explained in the right column.

RunProgram() Macro Code	Function
Dim Time As Range, Temp As Single	Creates a variable in which to store the overall program run time (Time) and a variable used to temporarily store the starting time (Temp).
Application.StatusBar = "(1/3) Checking Input Data"	Puts a message in the status bar (located in the lower left corner of the spreadsheet screen) to inform the user that the program is beginning the first of three steps and describes what will be done in that step.
Range("A1").Select Set Time = Range("Time") Time.ClearContents Temp = Timer	Stores the cursor off-screen (column A and row 1 are hidden on every worksheet). Initializes the Time variable and stores it in a cell named "Time." Deletes anything currently in that cell and stores the current time (Timer), which will be used as the program start time.
Application.ScreenUpdating = False Call SetRanges	Prevents the program from updating the screen after each step. This allows the program to run faster. Calls SetRanges() in mod01RunProgram to initialize the variables used by the program. (See Section 3.1.3.1 for more information about SetRanges()).
FullChemTable.Columns.AutoFit FullBioTable.Columns.AutoFit	Adjusts the sizes of the columns in the ChemTable and the BioTable.
ProgramName = ActiveWorkbook.Name NumChem = Application.CountA(ChemHeader) NumBioDataSets = Application.CountA(BioHeader) NumSamples = Application.CountA(ChemSample) NumSheets = Sheets.Count	Evaluates some of the parameters that will be needed in later procedures: ProgramName = name of spreadsheet, NumChem = no. of analytes in the chem table, NumBioDataSets = no. of sets of biological data, NumSamples = no. of samples in chem table, and NumSheets = no. of worksheets in spreadsheet.
Application.StatusBar = "(2/3) Comparing Chem and Bio Data Sets"	Updates the note in the status bar.
Call CompareData	Calls CompareData() in mod01RunProgram, which does the following:

RunProgram() Macro Code	Function
	<p>(1) Puts ChemData and BioData tables in survey, station, and sample order.</p> <p>(2) Trims leading and trailing blanks that might be in the data.</p> <p>(3) Sequentially compares survey, station, and sample names in the ChemData table to the survey, station, and sample names in the BioData table.</p> <p>(4) If any mismatches are found, calls DataProblemMessage(), which displays an error message and then ends the program.</p>
SaveSortIndicator = UCase(SaveSortIndicator) If SaveSortIndicator <> "Y" And SaveSortIndicator <> "N" Then SaveSortIndicator.Select MsgBox (" 'Save Sorted Data?' must contain Y or N. ") End End If	<p>Makes SaveSortIndicator an uppercase letter.</p> <p>Checks to see if the save sorted data question has a Y or N answer. If not, an error message is displayed and the program is stopped.</p>
AnovaName.ClearContents AnovaResults.ClearContents AnovaResultsHeader.ClearContents SumResultsTable.ClearContents SumResultsHeader.ClearContents RetainBio = "#N/A" RetainChem = "#N/A"	<p>Clears old data from the Anova Results and SumResults Tables.</p> <p>Removes all check marks from and shades the row of check boxes marked "Select one biological data set." and the column of check boxes marked "Deselect undesired analytes." on the [AnovaResults] page.</p>
For iChem = 1 To NumChem AnovaName(iChem) = ChemHeader(iChem) RetainChem(iChem) = "TRUE" Next iChem	Copies the analyte names from the ChemData Table into the Anova tables.
Sheets("AnovaResults").Select	Moves program activity to the [AnovaResults] page.
For iDataSet = 1 To NumBioDataSets	Initiates a series of calculations for each column of bio test results.
Application.StatusBar = "(3/3) Sorting Data and Calculating ANOVA for Dataset " & iDataSet & "/" & NumBioDataSets	Updates the note in the status bar to identify what bio data set is being evaluated.
FullSortedTable.ClearContents	Clears old data from the [SortedData] page.
RetainBio(iDataSet) = "FALSE"	Removes the shading from the check box above the set of bio data that will be tested with an analysis of variance.
Call SortData	<p>Calls SortData() in mod03SortData, which does the following:</p> <p>(1) Puts analyte names in every other column at the top of the data table.</p> <p>(2) Labels alternate columns in the second row of the data table as "Hit" and "NoHit" so that there is a pair for each</p>

RunProgram() Macro Code	Function
	<p>analyte.</p> <p>(3) Examines the data points for each analyte and puts the hit concentrations and the no-hit concentrations into the appropriately labeled columns.</p> <p>(4) Sorts each column of hit and no-hit data from low to high.</p>
If SaveSortIndicator = "Y" Then Call SaveSortedData	<p>If you want to save the sets of sorted hit and no-hit data, this calls SaveSortedData() in mod01RunProgram, which does the following:</p> <p>(1) Copies the table of data on the [SortedData] page.</p> <p>(2) Unhides the next available data storage worksheet.</p> <p>(3) Names the unhidden worksheet with the name of the bio data set.</p> <p>(4) Pastes the data set into the unhidden worksheet.</p>
Call Anova	<p>Calls Anova() in mod02Anova, which does the following:</p> <p>(1) Put zeroes in the "Totals by Significance Table."</p> <p>(2) Deletes old data from the [AnovaResults] page and enters the name of the bio data set that will be tested.</p> <p>(3) Uses the Analysis of Variance routine that is built into Microsoft Excel to calculate the Anova results for the hit and no-hit data for each analyte.</p> <p>(4) Based on the results of the ANOVA, assigns one of the following codes to each analyte:</p> <p>0 = No apparent difference in hit and no-hit distributions 0* = Significant differences at $p < 0.1$ 1 = Significant differences at $p < 0.05$ 1* = Significant differences at $p < 0.005$ 1** = Significant differences at $p < 0.0005$</p> <p>(5) Totals the number of each significance level for each analyte.</p>
Next iDataSet	Continues with the next bio data set.
Application.ScreenUpdating = False	Prevents the program from updating the screen after each step. This allows the program to run faster.
<p>If SaveSortIndicator = "Y" Then NumStart = NumBioDataSets + 1 Else NumStart = 1 End If</p> <p>For i = NumStart + 7 To NumSheets Sheets(i).Visible = False Next i</p>	<p>Determines where to start hiding the remaining unused data storage worksheets. If you do not save sorted data you start with the first worksheet after the 7 regular worksheets. If you save the sorted data you start with the first worksheet after the total of the 7 regular and the saved worksheets (one worksheet is saved for each bio data set).</p> <p>Hides the unused data storage worksheets.</p>

RunProgram() Macro Code	Function
Call FormatSortedTable	<p>Calls FormatSortedTable() in mod04FormatData, which does the following:</p> <p>(1) Merges the pairs of Hit/No-Hit header columns (each analyte has one column of hit data and one column of no-hit data) so that the pair of columns has a single header cell containing the name of the analyte.</p> <p>(2) Adjusts the columns to appropriate widths to deal with the widely varying length of the analyte names.</p>
Sheets("AnovaResults").Select AnovaResultsHeader.Columns.AutoFit Range("A1").Select	<p>Selects the page that will appear when the program ends.</p> <p>Adjusts the widths of the columns of Anova results.</p> <p>Stores the cursor off-screen (column A and row 1 are hidden on every worksheet).</p>
Application.ScreenUpdating = True Application.StatusBar = False	<p>Restores screenupdating so changes made during the running of the program will now show up on screen.</p> <p>Deletes the message from the status bar.</p>
Time = Timer – Temp MsgBox (" Calculations Completed ")	<p>Subtracts the initial time (Temp) from the current time (Timer) to get the program run time.</p> <p>Puts message on screen to inform the user that the program has ended.</p>

3.4.2 mod01RunProgram

This module contains 8 macros:

RunProgram controls the overall program by calling the macros that carry out the necessary sorting, testing, and calculating routines.

SetRanges initializes the variables by setting them equal to some of the relevant previously-declared ranges.

CompareData compares the chem table survey, station, and sample data to the corresponding data in the bio table. If they are not the same the program stops and an error message is displayed.

DataProblemMessage displays the error message called by CompareData.

SaveSortedData copies data temporarily stored on the [SortedData] page, and puts it onto a separate page after the [AnovaResults] page.

DeleteStoredData deletes the data saved by the above macro, renames the page, and hides the empty page.

CopySortedData copies the sorted data table from the FPMData spreadsheet and pastes it into this FPMAnova spreadsheet.

CompileDataSet constructs the [FinalDataset] from the bio data set and analytes that are selected on the [AnovaResults] page.

The code and comments for this module are shown below.

```

'   File:   FPMAnova.xls

'   This is the second of three spreadsheets used for implementing the
'   Floating Percentile Method (FPM) developed by Teresa Michelsen of
'   Avocet Consulting. The FPM is used to assess sediment toxicity
'   and chemical composition data to develop sediment quality guidelines.

'   FPMAnova.xls:
'   (1) Accepts screened analytical data from FPMDData.xls;
'   (2) Requires one or more sets of bio data with the same survey,
'       station, and sample IDs as the analytical data;
'   (3) Sorts the data by analyte data into hit and no-hit groups;
'   (4) Stores the sorted hit/no-hit data in another workbook, if desired; and
'   (5) Calculates analysis of variance on the hit versus no-hit data.

'   This spreadsheet was developed by Michael R. Anderson,
'   Oregon Department of Environmental Quality.

'   Last edited: July 29, 2008

'   Names shown in [Square Brackets] in the comments are names of worksheets.

Option Private Module
Option Explicit

Public AnovaResults As Range, AnovaResultsHeader As Range
Public AnovaF As Range, AnovaFcrit As Range
Public AnovaName As Range, AnovaTable As Range
Public BioCheckBoxes As Range, BioIDTable As Range, BioIDHeader As Range
Public BioSample As Range, BioStation As Range, BioSurvey As Range
Public BioTable As Range, BioHeader As Range, ChemCheckBoxes As Range
Public ChemHeader As Range, ChemIDHeader As Range, ChemIDTable As Range
Public ChemSample As Range, ChemStation As Range, ChemSurvey As Range
Public ChemTable As Range, ChemTableNoHeader As Range, ChemUnits As Range
Public FinalBioName As Range, FinalDataHeader As Range, FinalDataTable As Range
Public FinalHitNoHit As Range, FinalIDTable As Range
Public FullChemTable As Range, FullBioTable As Range, FullFinalDataTable As Range
Public FullSortedTable As Range, NumData As Range
Public Pvalue As Range, RetainBio As Range, RetainChem As Range
Public SaveSortIndicator As Range, SortedHeaderHitNoHit As Range
Public SortedHeaderName As Range, SortedTable As Range
Public SumResultsHeader As Range, SumResultsTable As Range
Public ToolBarStorage As Range, VersionName As Range

Public i As Integer, iChem As Integer, iCount As Integer
Public iColumn As Integer, iDataSet As Integer, iSample As Integer
Public NumChem As Integer, NumBioDataSets As Integer
Public NumHit As Integer, NumIndet As Integer, NumNoHit
Public NumPage As Integer, NumSamples As Integer, NumStart As Integer
Public NumSheets As Integer, NumStations As Integer

Public AnovaRange As Object, Workrange As Object

Public ProgramName As String

Public ColWid As Single

```

```

' *****
' RunProgram() controls the overall program by calling the macros
' that carry out the necessary sorting, testing, and calculating routines.
' *****

Sub RunProgram()

Dim Time As Range, Temp As Single

Application.StatusBar = "(1/3) Checking Input Data"
Range("A1").Select
Set Time = Range("Time")
Time.ClearContents
Temp = Timer

Application.ScreenUpdating = False

Call SetRanges
FullChemTable.Columns.AutoFit
FullBioTable.Columns.AutoFit

' Collect basic information needed in other macros.

ProgramName = ActiveWorkbook.Name
NumChem = Application.CountA(ChemHeader)
NumBioDataSets = Application.CountA(BioHeader)
NumSamples = Application.CountA(ChemSample)
NumSheets = Sheets.Count

Application.StatusBar = "(2/3) Comparing Chem and Bio Data Sets"
Call CompareData

' Make sure that the save sorted data question has a Y or N answer.

SaveSortIndicator = UCase(SaveSortIndicator)
If SaveSortIndicator <> "Y" And SaveSortIndicator <> "N" Then
    SaveSortIndicator.Select
    MsgBox (" 'Save Sorted Data?' must contain Y or N. ")
End
End If

' Clear unneeded data remaining from previous tests

AnovaName.ClearContents
AnovaResults.ClearContents
AnovaResultsHeader.ClearContents
SumResultsTable.ClearContents
SumResultsHeader.ClearContents
RetainBio = "#N/A"
RetainChem = "#N/A"

' Copy analyte names from the ChemData Table into the Anova tables

For iChem = 1 To NumChem
    AnovaName(iChem) = ChemHeader(iChem)
    RetainChem(iChem) = "TRUE"
Next iChem

```



```

'    Carry out the calculations for each column of bio test results

Sheets("AnovaResults").Select

For iDataSet = 1 To NumBioDataSets
    Application.StatusBar = "(3/3) Sorting Data and Calculating ANOVA for Dataset " &
iDataSet & "/" & NumBioDataSets
    FullSortedTable.ClearContents
    RetainBio(iDataSet) = "FALSE"
    Call SortData
    If SaveSortIndicator = "Y" Then Call SaveSortedData
    Call Anova
Next iDataSet

Application.ScreenUpdating = False

'    Hide unused sorted data sheets

If SaveSortIndicator = "Y" Then
    NumStart = NumBioDataSets + 1
Else
    NumStart = 1
End If

For i = NumStart + 7 To NumSheets
    Sheets(i).Visible = False
Next i

Call FormatSortedTable

Sheets("AnovaResults").Select
AnovaResultsHeader.Columns.AutoFit
Range("A1").Select

Application.ScreenUpdating = True
Application.StatusBar = False

Time = Timer - Temp
MsgBox ("    Calculations Completed    ")

End Sub

'    *****
'    SetRanges() initiates the variables by associating
'    them with the previously-declared ranges.
'    *****

Sub SetRanges()

Set BioCheckBoxes = Range("BioCheckBoxes")
Set FinalBioName = Range("FinalBioName")
Set FinalDataHeader = Range("FinalDataHeader")
Set FinalDataTable = Range("FinalDataTable")
Set FinalHitNoHit = Range("FinalHitNoHit")
Set FinalIDTable = Range("FinalIDTable")
Set AnovaResults = Range("AnovaResults")
Set AnovaResultsHeader = Range("AnovaResultsHeader")
Set AnovaF = Range("AnovaF")
Set AnovaFcrit = Range("AnovaFcrit")
Set AnovaName = Range("AnovaName")

```

```

Set AnovaTable = Range("AnovaTable")
Set BioIDHeader = Range("BioIDHeader")
Set BioIDTable = Range("BioIDTable")
Set BioSample = Range("BioSample")
Set BioStation = Range("BioStation")
Set BioSurvey = Range("BioSurvey")
Set BioHeader = Range("BioHeader")
Set BioTable = Range("BioTable")
Set ChemCheckBoxes = Range("ChemCheckBoxes")
Set ChemHeader = Range("ChemHeader")
Set ChemIDHeader = Range("ChemIDHeader")
Set ChemIDTable = Range("ChemIDTable")
Set ChemSample = Range("ChemSample")
Set ChemStation = Range("ChemStation")
Set ChemSurvey = Range("ChemSurvey")
Set ChemTable = Range("ChemTable")
Set ChemTableNoHeader = Range("ChemTableNoHeader")
Set ChemUnits = Range("ChemUnits")
Set FullChemTable = Range("FullChemTable")
Set FullBioTable = Range("FullBioTable")
Set FullFinalDataTable = Range("FullFinalDataTable")
Set FullSortedTable = Range("FullSortedTable")
Set NumData = Range("NumData")
Set Pvalue = Range("Pvalue")
Set RetainBio = Range("RetainBio")
Set RetainChem = Range("RetainChem")
Set SaveSortIndicator = Range("SaveSortIndicator")
Set SortedHeaderHitNoHit = Range("SortedHeaderHitNoHit")
Set SortedHeaderName = Range("SortedHeaderName")
Set SortedTable = Range("SortedTable")
Set SumResultsHeader = Range("SumResultsHeader")
Set SumResultsTable = Range("SumResultsTable")
Set ToolBarStorage = Range("ToolBarStorage")
Set VersionName = Range("VersionName")

```

End Sub

```

' *****
' CompareData() compares the chem table survey, station, and
' sample data to the corresponding data in the bio table.
' If they are not the same the program stops and an error
' message is displayed.
' *****

```

Sub CompareData()

```

' Make sure that the data in the ChemData and BioData tables are
' listed in the same order so comparisons can be made. This sorts each
' table into ascending order by survey, station, and sample.

```

```

FullBioTable.Sort _
    Key1:=BioSurvey, Order1:=xlAscending, _
    Key2:=BioStation, Order1:=xlAscending, _
    Key3:=BioSample, Order1:=xlAscending, _
    Header:=xlYes, _
    Orientation:=xlTopToBottom

```

```

ChemTableNoHeader.Sort _
    Key1:=ChemTableNoHeader(1, 1), Order1:=xlAscending, _
    Key2:=ChemTableNoHeader(1, 2), Order1:=xlAscending, _
    Key3:=ChemTableNoHeader(1, 3), Order1:=xlAscending, _
    Header:=xlNo, _

```

```

        Orientation:=xlTopToBottom

For i = 1 To NumSamples

'   In one of my early tests I kept getting error messages even though
'   I thought the survey, station, and sample data were clearly the same.
'   It turned out that the bio data all had trailing blank spaces, which
'   made them unequal to the chem entries. The Trim function removes
'   leading and trailing blank spaces to avoid this problem.

    BioSurvey(i) = Trim(BioSurvey(i))
    BioStation(i) = Trim(BioStation(i))
    BioSample(i) = Trim(BioSample(i))
    ChemSurvey(i) = Trim(ChemSurvey(i))
    ChemStation(i) = Trim(ChemStation(i))
    ChemSample(i) = Trim(ChemSample(i))

    If BioSurvey(i) <> ChemSurvey(i) Then Call DataProblemMessage
    If BioStation(i) <> ChemStation(i) Then Call DataProblemMessage
    If BioSample(i) <> ChemSample(i) Then Call DataProblemMessage

Next i

End Sub

' *****
'   DataProblemMessage() displays the error message called by CompareData().
' *****

Sub DataProblemMessage()

Dim Msg, Style, Title, Response
Msg = "The BioData survey, station, and sample identifiers do not match all of the" & Chr(13)
& _
"    corresponding ChemData survey, station, and sample identifiers." & Chr(13) & _
"    Do you want to continue with the ANOVA calculations?"
Style = vbYesNoCancel
Title = "Possible Data Mismatch"
Response = MsgBox(Msg, Style, Title)

If Response <> vbYes Then End

End Sub

' *****
'   SaveSortedData() copies data temporarily stored on the [SortedData]
'   page, and puts it onto a separate page after the [AnovaResults] page.
' *****

Sub SaveSortedData()

    Sheets("SortedData").Select
    Range(FullSortedTable(1, 1), FullSortedTable(NumSamples + 2, 2 * NumChem)).Select
    Selection.Copy
    Range("A1").Select

    Sheets(7 + iDataSet).Visible = True
    Sheets(7 + iDataSet).Select
    Sheets(7 + iDataSet).Name = BioHeader(iDataSet)
    Range("I3") = BioHeader(iDataSet)
    Range(Cells(5, 3), Cells(4 + NumSamples + 2, 2 + 2 * NumChem)).Select

```

```

ActiveSheet.Paste

Application.CutCopyMode = False
Range("A1").Select

End Sub

' *****
' DeleteStoredData() deletes the data saved by the above
' macro, renames the page, and hides the empty page.
' *****

Sub DeleteStoredData()

Application.ScreenUpdating = False

Call SetRanges
NumChem = Application.CountA(ChemHeader)
NumBioDataSets = Application.CountA(BioHeader)
NumSamples = Application.CountA(ChemSample)
NumSheets = Sheets.Count

For iDataSet = 1 To NumSheets - 7
    If Sheets(7 + iDataSet).Visible = False Then Sheets(7 + iDataSet).Visible = True
    Sheets(7 + iDataSet).Select
    Sheets(7 + iDataSet).Name = iDataSet
    Range("I3") = ""
    Range(Cells(5, 3), Cells(5 + NumSamples + 2, 3 + 2 * NumChem)).Select
    Selection.ClearContents
    Range("A1").Select
    Sheets(7 + iDataSet).Visible = False
Next iDataSet

If iDataSet < (NumSheets - 7) Then
    For i = iDataSet To (NumSheets - 7)
        Sheets(7 + i).Visible = False
    Next i
End If

Sheets("ControlScreen").Select
Application.ScreenUpdating = True

End Sub

' *****
' CopySortedData() copies the sorted data table from the FPMData
' spreadsheet and pastes it into this FPMANOVA spreadsheet.
' *****

Sub CopySortedData()

Dim SourceName As String, DestinationName As String
Dim DefaultSourceName As Range
Set DefaultSourceName = Range("DefaultSourceName")

Range("A1").Select
Application.ScreenUpdating = False

Call SetRanges
ProgramName = ActiveWorkbook.Name

```

```

On Error GoTo EndThis
SourceName = DefaultSourceName
DestinationName = ProgramName

Windows(SourceName).Activate
Range("FullDataTable").Copy

Windows(DestinationName).Activate
Sheets("ChemData").Select
Range("FullChemTable").PasteSpecial Paste:=xlPasteValues
Application.CutCopyMode = False

Range("A1").Select

FullChemTable.Columns.AutoFit

Application.ScreenUpdating = True
Exit Sub

EndThis:

MsgBox ("          The name you entered did not work." & Chr(13) & _
"          Make sure that the source file is open" & Chr(13) & _
"and in the same folder as this spreadsheet," & Chr(13) & _
"          or check the spelling and try again.")

End Sub

' *****
' CompileDataSet() constructs the [FinalDataset] from the bio data
' set and analytes that are selected on the [AnovaResults] page.
' *****

Sub CompileDataSet()

Application.ScreenUpdating = False

Call SetRanges

NumChem = Application.CountA(ChemHeader)
NumBioDataSets = Application.CountA(BioHeader)
NumSamples = Application.CountA(ChemSample)
Call ClearFinalTable

' Check to make sure that only one bio data set has been selected
' and that at least one analyte has been selected.

If Application.CountIf(RetainBio, "TRUE") = 0 Then
    Sheets("AnovaResults").Select
    BioCheckBoxes.Select
    Application.ScreenUpdating = True
    MsgBox ("One biological data set must be selected.")
    End
ElseIf Application.CountIf(RetainBio, "TRUE") > 1 Then
    Sheets("AnovaResults").Select
    BioCheckBoxes.Select
    Application.ScreenUpdating = True
    MsgBox ("Only one biological data set can be selected.")
    End
ElseIf Application.CountIf(RetainChem, "TRUE") = 0 Then
    Sheets("AnovaResults").Select
    ChemCheckBoxes.Select

```

```

Application.ScreenUpdating = True
MsgBox ("At least one analyte must be selected.")
End
End If

' Examine each column of bio data on the [AnovaResults] worksheet.
' When the one column with a checked box is found ("True"), copy
' the corresponding column from the [BioData] worksheet and paste it
' into the Hit/No-Hit column of the table on [FinalDataSet]. Copy the
' Survey, Station, and Sample information from [BioData] and paste it
' into the table on [FinalDataSet] as well.

For iDataSet = 1 To NumBioDataSets
    If RetainBio(iDataSet) = "True" Then
        FinalBioName = BioHeader(iDataSet)

        BioIDTable.Copy
        FinalIDTable.PasteSpecial Paste:=xlPasteValues
        Application.CutCopyMode = False

        Range(BioTable(1, iDataSet), BioTable(NumSamples, iDataSet)).Copy
        Range(FinalHitNoHit(1), FinalHitNoHit(NumSamples)).PasteSpecial Paste:=xlPasteValues
        Application.CutCopyMode = False

        Exit For
    End If
Next iDataSet

' Examine each analyte name in the list on the [AnovaResults] worksheet.
' When one with a checked box is found ("True"), copy the relevant
' column from the [ChemData] worksheet and paste it into the first available
' column of analytes on the [FinalDataSet] worksheet. Continue until all
' checked analytes have been copied.

iCount = 0
For iChem = 1 To NumChem
    If RetainChem(iChem) = "True" Then
        iCount = iCount + 1
        Range(FullChemTable(1, 3 + iChem), FullChemTable(2 + NumSamples, 3 + iChem)).Copy
        Range(FinalDataTable(1, iCount), FinalDataTable(2 + NumSamples, iCount)).PasteSpecial
        Paste:=xlPasteValues
        Application.CutCopyMode = False
    End If
Next iChem

Sheets("FinalDataSet").Select
FinalDataTable.Select
Call CenterText

For i = 1 To 100
    If FinalDataHeader(i).ColumnWidth < 10 Then FinalDataHeader(i).ColumnWidth = 10
Next i

Range("A1").Select
Application.ScreenUpdating = True

End Sub

```

This is the end of FPMAnova.xls mod01RunProgram.
--

3.4.3 mod02Anova

This module contains only 1 macro, **Anova**. The code and comments are shown below.

```
Option Private Module
Option Explicit
```

```
' *****
' Anova() performs an analysis of variance on the Hit and No-Hit data
' for each analyte and returns one of the following results:
'
' 0   = This analyte showed no apparent difference in its hit and
'       no-hit distributions for this hit/no-hit definition.
' 0*  = This analyte showed significant differences in its hit and
'       no-hit distributions for this hit/no-hit definition (p < 0.1).
' 1   = Same as above except that p < 0.05
' 1*  = Same as above except that p < 0.005
' 1** = Same as above except that p < 0.0005
' *****
```

```
Sub Anova()
```

```
' Put zeroes in the 'Totals by Significance Table' before starting to count

For i = 1 To 5
    SumResultsTable(i, iDataSet) = 0
Next i

' For each analyte, perform ANOVA on the hit vs. nohit data and
' note the significance level group to which the result belongs.

Application.ScreenUpdating = True

For iChem = 1 To NumChem

    AnovaTable.ClearContents
    AnovaResultsHeader(iDataSet) = BioHeader(iDataSet)
    SumResultsHeader(iDataSet) = BioHeader(iDataSet)

    Set AnovaRange = Range(SortedHeaderHitNoHit(2 * iChem - 1),
SortedTable(NumData(iDataSet), 2 * iChem))
```

The command below initiates the Analysis of Variance routine that is built into Microsoft Excel and uses it to calculate the Anova results for the hit and no-hit data.

```
Application.Run "ATPVBAEN.XLA!Anova1", AnovaRange, AnovaTable, "C", True, 0.1

If AnovaF > AnovaFcrit Then
    If Pvalue < 0.0005 Then
        AnovaResults(iChem, iDataSet) = "1**"
        SumResultsTable(1, iDataSet) = SumResultsTable(1, iDataSet) + 1
    ElseIf Pvalue < 0.005 Then
        AnovaResults(iChem, iDataSet) = "1*"
        SumResultsTable(2, iDataSet) = SumResultsTable(2, iDataSet) + 1
    ElseIf Pvalue < 0.05 Then
        AnovaResults(iChem, iDataSet) = "1"
        SumResultsTable(3, iDataSet) = SumResultsTable(3, iDataSet) + 1
    ElseIf Pvalue < 0.1 Then
        AnovaResults(iChem, iDataSet) = "0**"
```

```
        SumResultsTable(4, iDataSet) = SumResultsTable(4, iDataSet) + 1
    End If
Else
    AnovaResults(iChem, iDataSet) = "0"
    SumResultsTable(5, iDataSet) = SumResultsTable(5, iDataSet) + 1
End If

Next iChem

Application.ScreenUpdating = False

End Sub
```

This is the end of FPMAnova.xls mod02Anova.

3.4.4 mod03Sort Data

This module contains only 1 macro, **SortData**. The code and comments are shown below.

```
Option Private Module
Option Explicit

' *****
' SortData() uses the information from [BioData] to create a new table
' on the [SortedData] worksheet in which the analytical data are separated
' into Hit and No-Hit groups for each analyte.
' *****

Sub SortData()

For iChem = 1 To NumChem      'This is the column number

    SortedHeaderName(2 * iChem - 1) = ChemHeader(iChem)
    SortedHeaderHitNoHit(2 * iChem - 1) = "Hit"
    SortedHeaderHitNoHit(2 * iChem) = "NoHit"

    For iSample = 1 To NumSamples      'This is the row number

' If there is an entry in the ChemTable, determine if it is a Hit or NoHit
' and copy the entry into the appropriate column in the SortedTable

        If ChemTable(iSample, iChem) <> "" Then
            If BioTable(iSample, iDataSet) = 1 Then SortedTable(iSample, 2 * iChem - 1) =
ChemTable(iSample, iChem)
            If BioTable(iSample, iDataSet) = 0 Then SortedTable(iSample, 2 * iChem) =
ChemTable(iSample, iChem)
        End If

    Next iSample
Next iChem

' Sort each column of chem data from low to high.

For iColumn = 1 To 2 * NumChem

    Range(SortedTable(1, iColumn), SortedTable(NumSamples, iColumn)).Sort _
        Key1:=SortedTable(1, iColumn), Order1:=xlAscending, _
        Header:=xlNo, _
        Orientation:=xlTopToBottom

Next iColumn

End Sub
```

This is the end of FPMAnova.xls mod03Sort Data.

3.4.5 mod04Format Pages

This module contains 11 macros, most of which are used to delete data from tables or format data in tables.

Auto_Open sets up the appearance of the workbook upon opening.

SetUpScreen closes toolbars and sets up the worksheets in preparation for using the program. It's called from the Auto_Open procedure.

CenterText centers selected text and adjusts columns to appropriate widths.

FormatSortedTable merges pairs of Hit/No-Hit header columns and adjusts them to appropriate widths.

ClearAllTables deletes entries from all tables. It's called from a button on the [ControlScreen] page.

ClearChem deletes entries from the [ChemData] page. It's called from a button on the [ControlScreen] page.

ClearBio deletes entries from the [BioData] page. It's called from a button on the [ControlScreen] page.

ClearSorted deletes entries from the [SortedData] page. It's called from a button on the [ControlScreen] page.

ClearAnova deletes entries from the [AnovaResults] page. It's called from a button on the [ControlScreen] page.

ClearFinalTable deletes entries from the [FinalDataSet] page. It's called from the ClearAllTables macro.

Auto_Close restores original toolbars, menus, etc. It works automatically on closing.

The code for these macros is shown below.

```
Option Private Module
Option Explicit

' *****
'   Auto_Open() sets up the appearance of the workbook upon opening.
'   *****

Sub Auto_Open()

Application.ScreenUpdating = False
Application.StatusBar = False

Call SetRanges
Call SetUpScreen

'   If the program is revised change the version date below.

VersionName = "FPMAnova.xls Version 072908"

Sheets("ControlScreen").Activate
Application.ScreenUpdating = True

End Sub
```

```

' *****
' SetUpScreen() closes toolbars and sets up the worksheets
' in preparation for using the program. It's called from
' the Auto_Open procedure.
' *****

Sub SetUpScreen()

Dim Bar As Object, BarCount As Integer

' This sets each worksheet back to a standard top-left
' orientation with the cursor hidden off screen and hides
' the sheet that's used to store reference information.

For i = 1 To 7
    Sheets(i).Activate
    ActiveWindow.DisplayHeadings = False
    Application.DisplayCommentIndicator = xlCommentIndicatorOnly
    ActiveWindow.ScrollColumn = 1
    ActiveWindow.ScrollRow = 1
    Range("A1").Select
Next i

Sheets(7).Visible = False

' To create more space, hide the Formula Bar and the Comment Indicator

Application.DisplayCommentIndicator = False
Application.DisplayFormulaBar = False

' Keep track of what toolbars are open so they can be reopened
' before exiting. Names are stored on the hidden worksheet [Names].
' Then hide the open toolbars to create more space on the screen.

ToolBarStorage.ClearContents
BarCount = 0

For Each Bar In Toolbars
    If Bar.Visible = True Then
        BarCount = BarCount + 1
        ToolBarStorage(BarCount) = Bar.Name
        Bar.Visible = False
    End If
Next Bar

End Sub

' *****
' CenterText() centers selected text and adjusts columns
' to appropriate widths.
' *****

Sub CenterText()

With Selection
    .HorizontalAlignment = xlCenter
    .Columns.AutoFit
End With

Range("A1").Select

End Sub

```

```
' *****
' FormatSortedTable() merges pairs of Hit/No-Hit header
' columns and adjusts them to appropriate widths.
' *****
```

```
Sub FormatSortedTable()
```

```
Sheets("SortedData").Activate
```

```
SortedHeaderName.Select
```

```
With Selection
```

```
    .MergeCells = False
```

```
    .HorizontalAlignment = xlLeft
```

```
    .Columns.AutoFit
```

```
End With
```

```
For i = 1 To 200 Step 2
```

```
ColWid = SortedHeaderName(i).ColumnWidth
```

```
If ColWid < 20 Then ColWid = 20
```

```
SortedHeaderName(i).ColumnWidth = ColWid / 2
```

```
SortedHeaderName(i + 1).ColumnWidth = ColWid / 2
```

```
Range(SortedHeaderName(i), SortedHeaderName(i + 1)).Select
```

```
With Selection
```

```
    .HorizontalAlignment = xlCenter
```

```
    .MergeCells = True
```

```
End With
```

```
Next i
```

```
Range("A1").Select
```

```
End Sub
```

```
' *****
' ClearAllTables() deletes entries from all tables.
' It's called from a button on the [ControlScreen] page.
' *****
```

```
Sub ClearAllTables()
```

```
Dim Msg, Style, Title, Response
```

```
Msg = " Are you sure that you want to clear ALL of the tables? "
```

```
Style = vbYesNoCancel
```

```
Title = "Clear All Tables"
```

```
Response = MsgBox(Msg, Style, Title)
```

```
If Response = vbYes Then
```

```
    Call ClearChem
```

```
    Call ClearBio
```

```
    Call ClearSorted
```

```
    Call ClearAnova
```

```
    Call DeleteStoredData
```

```
    Call ClearFinalTable
```

```
Else
```

```
    Exit Sub
```

```
End If
```

```
End Sub
```

```

' *****
' ClearChem() deletes entries from the [ChemData] page.
' It's called from a button on the [ControlScreen] page.
' *****

Sub ClearChem()
    Call SetRanges
    ChemIDTable.ClearContents
    ChemTable.ClearContents
    ChemHeader.ClearContents
    ChemUnits.ClearContents
End Sub

' *****
' ClearBio() deletes entries from the [BioData] page.
' It's called from a button on the [ControlScreen] page.
' *****

Sub ClearBio()
    Call SetRanges
    BioIDTable.ClearContents
    BioTable.ClearContents
    BioHeader.ClearContents
End Sub

' *****
' ClearSorted() deletes entries from the [SortedData] page.
' It's called from a button on the [ControlScreen] page.
' *****

Sub ClearSorted()
    Call SetRanges
    FullSortedTable.ClearContents
End Sub

' *****
' ClearAnova() deletes entries from the [AnovaResults] page.
' It's called from a button on the [ControlScreen] page.
' *****

Sub ClearAnova()
    Call SetRanges
    AnovaName.ClearContents
    AnovaResults.ClearContents
    AnovaResultsHeader.ClearContents
    SumResultsTable.ClearContents
    SumResultsHeader.ClearContents
    RetainBio = "#N/A"
    RetainChem = "#N/A"
End Sub

```

```
' *****
' ClearFinalTable() deletes entries from the [FinalDataSet] page.
' It's called from the ClearAllTables macro.
' *****
```

```
Sub ClearFinalTable()
    Call SetRanges
    FinalIDTable.ClearContents
    FinalDataTable.ClearContents
    FinalBioName.ClearContents
    FinalHitNoHit.ClearContents
End Sub
```

```
' *****
' Auto_Close() restores original toolbars, menus, etc.
' Works automatically on closing.
' *****
```

```
Sub Auto_Close()

    Call SetRanges
    On Error Resume Next
    Application.ScreenUpdating = False
    Sheets("ControlScreen").Select

    For i = 1 To Application.CountA(ToolBarStorage)
        Toolbars(ToolBarStorage(i).Value).Visible = True
    Next i

    Application.DisplayFormulaBar = True
    Application.DisplayCommentIndicator = True

End Sub
```

This is the end of FPMAnova.xls mod04Format Pages.
--

This is the end of macro code for the FPMAnova.xls Worksheet.

3.5 Macro Code for FPMCalc.xls

3.5.1 Summary

FPMAnova.xls contains 32 Microsoft Excel® Visual Basic macros that perform specific functions for the spreadsheet. The macros are stored in ten modules. The code from each module along with some explanatory text is listed in Sections 3.5.2 to 3.5.11. A password is required to access the modules in the spreadsheet.

3.5.1.1 Macros Activated When Spreadsheet Opens or Closes

MacroName() Module Containing Macro	Macro Action
Auto_Open() mod10FormatData	When you open the spreadsheet this macro (1) calls the macro SetRanges() in mod01RunProgram to initialize all of the variables, (2) calls the macro SetUpScreen() in mod10FormatPages to close unneeded toolbars, and (3) resets all worksheets to the standard top-left orientation.
Auto_Close() mod10FormatData	When you close the spreadsheet this macro restores the toolbars that were closed when the program opened and resets the formats of the worksheets.

3.5.1.2 Macros Activated from Buttons on Worksheets

MacroName() Module Containing Macro	Button that Activates Macro [Worksheet with Button]	Macro Action
CopyChemData() mod09CountCopyData	Copy Anova Data [ControlScreen]	Copies data from [FinalDataSet] page of the FPMAnova.xls spreadsheet and pastes it into the [DataTable] page of the FPMCalc.xls spreadsheet.
ClearAllTables() mod01RunProgram	Clear All Worksheets [ControlScreen]	Deletes entries from FullDataTable. Calls ClearTables() and ClearTestTable() in mod01RunProgram to delete entries from the remaining tables.
ClearTables() mod01RunProgram	Clear All Worksheets Except Data & Storage Tables [ControlScreen]	Deletes entries from the tables that store results so that data from a previous calculation will not interfere. Calls ClearTestTable() in mod01RunProgram to delete entries from that table.
RunProgram() mod01RunProgram	Calculate Floating Percentiles [ControlScreen]	See Section 3.5.1.3
ShowReliabilityDefs() mod09CountCopyData	Definitions of Reliability Measures [Criteria]	Displays the ReliabilityDefs form. See Section 2.4.3.2
ShowHideCriteriaRows() Mod10FormatData	Show / Hide Additional Calculation Details [Criteria]	Shows or hides the extra rows of data on the [Criteria] page.

MacroName() Module Containing Macro	Button that Activates Macro [Worksheet with Button]	Macro Action
ShowCriteriaDefs() mod09CountCopyData	Description of Calculation Details [Criteria]	Displays the CriteriaDefs form. See Section 2.4.3.1

3.5.1.3 Macros that Control the Main Function of the Program

When you click on the “**Calculate Floating Percentiles**” button, which is located on the [ControlScreen] worksheet, you start the RunProgram() macro in the mod01RunProgram module. This macro performs some basic functions on its own and also calls other macros that each carry out part of the overall process. In the table below the RunProgram() macro code is shown in the left column and the steps that are carried out are explained in the right column.

RunProgram() Macro Code	Function
Range("A1").Select	Stores the cursor off-screen (column A and row 1 are hidden on every worksheet).
Call SetRanges	Calls SetRanges() in mod01RunProgram to initialize the variables used by the program. (See Section 3.1.3.1 for more information about SetRanges()).
VersionName = "FPMCalc.xls Version 120608"	Puts program version on [ControlScreen]
Time = ""	Clears the time from the previous run.
DataSummary.ClearContents	Clears all entries from the Data Summary Table on [ControlScreen].
NumSteps = Int(1 + (FNFinal - FNInitial) / FNInterval)	Determines the number of false negative levels to calculate based on the requested initial, final, and interval values.
Temp = Timer	Stores the current time (Timer), which will be used as the program start time.
Call Count	Calls Count() in mod09CountCopyData, which does the following: (1) Makes note of the spreadsheet's current name. (2) Counts the following items for later use. NumChem = No. of analytes NumData = No. of data points NumSamples = No. of samples NumIndet = No. of bio results considered indeterminate NumNoHits = No. of bio results considered no-hits NumHits = No. of bio results considered hits
Application.ScreenUpdating = False	Prevents the program from updating the screen after each step. This allows the program to run faster.
Call ClearTables	Calls ClearTables() in mod01RunProgram, which does the following: Deletes entries from the tables that store results so that data from a previous calculation will not interfere. Calls ClearTestTable() in mod01RunProgram to delete entries from that table.
Call InputCheck	Calls InputCheck() in mod01RunProgram, which does the

RunProgram() Macro Code	Function
	<p>following:</p> <p>(1) Checks to see if all of the required Yes/No answers have been provided and if reasonable values for the range of %False Negatives have been entered in the "Program Settings" table on the [ControlScreen] page.</p> <p>(2) If not, an error message is displayed and the program is stopped.</p>
Application.StatusBar = "(1/6) Creating Distributions"	Puts a message in the status bar (located in the lower left corner of the spreadsheet screen) to inform the user that the program is beginning the first of six steps and describes what will be done in that step.
Call CreateDistributions	<p>Calls CreateDistributions() in mod02CreateDistributions, which does the following:</p> <p>(1) Copies the analyte names from [DataTable] to [Distributions].</p> <p>(2) Copies data from [DataTable] and sorts it into columns by analyte in the table on [Distributions].</p> <p>(3) Sorts each column of data from low to high.</p>
Application.StatusBar = "(2/6) Summarizing Data"	Updates the note in the status bar.
Call SummarizeData	<p>Calls SummarizeData() in mod03SummarizeData, which does the following:</p> <p>(1) Sorts the Hit/No-Hit entries on [Data Table] from low to high, which orders them as indeterminates (-1), no-hits (0), and hits(1).</p> <p>(2) Counts the number of data points and finds the maximum and minimum for each analyte.</p> <p>(3) Uses the min and max concentrations of each analyte to evaluate the initial increments used for the floating percentile calculations.</p> <p>(4) Estimates the number of iterations necessary to reach the desired precision.</p> <p>(5) Counts the hits and no-hits and finds the maximum hit and no-hit concentrations.</p> <p>(6) Assigns an AET to each analyte. If the option has been selected, outliers are omitted from consideration when assigning the AETs.</p>
Application.StatusBar = "(3/6) Calculating Percentiles"	Updates the note in the status bar.
Call CalculatePercentiles	Calls CalculatePercentiles() in mod04CalculatePercentiles, which creates a percent distribution list for the concentrations of each analyte based on the raw data distributions on the [Distributions] page. Stores the results on the [Percentiles] page.

RunProgram() Macro Code	Function
Application.StatusBar = "(4/6) Calculating Error Levels"	Updates the note in the status bar.
Call ErrorCalculations	<p>Calls ErrorCalculations() in mod05ErrorCalculations, which does the following:</p> <p>(1) Compares each row of percentile concentrations to analyte concentrations and determines if the percentile concentrations would correctly predict a hit or a no-hit.</p> <p>(2) Counts the number of true and false hits and no-hits for each percentile.</p> <p>(3) Uses the true/false hit/no-hit data to calculate a series of reliability parameters for each percentile. See Section 2.4.3.2 for definitions of the reliability parameters.</p>
Application.StatusBar = "(5/6) Selecting Criteria"	Updates the note in the status bar.
Call SelectCriteria	<p>Calls SelectCriteria() in mod06SelectCriteria, which does the following:</p> <p>(1) Puts analyte names on the [Criteria] page.</p> <p>(2) Finds the data percentiles closest to the specified percent false negatives and copies the concentrations from those percentiles to the [Criteria] page where they will be used as the initial concentrations in the FPM calculations.</p>
Application.StatusBar = "(6/6) Calculating FloatingPercentiles"	Updates the note in the status bar.
Call FloatingPercentilesPass1 <u>Note:</u> FloatingPercentilesPass1() includes a large amount of explanatory text that is not included in this table. For additional information please see Error! Reference source not found.on page Error! Bookmark not defined..	<p>Calls FloatingPercentilesPass1() in mod07FloatingPercentilesPass1, which does the following for each requested %FN:</p> <p>(1) Determines the initial increment that will be used for each analyte during the FPM calculations.</p> <p>(2) Copies the initial concentrations for the analytes, which were determined previously by SelectCriteria(), and pastes them into the row where they will be modified until the final concentrations are reached.</p> <p>(3) Calls CountFalsePos(2), which counts the #FP for each of the concentrations in the above step.</p> <p>(4) Calls RankChem(), which sorts the analytes in order of #FPs from highest to lowest, assigns a rank to each analyte starting with 1 for the highest #FPs, and then puts the analytes back in alphabetical order.</p> <p>(5) Finds the analyte with the most #FPs, raises its initial concentration by the initial increment determined in step(1), calls CountFalsePos(2) to recount the #FPs for each analyte, and calls CountFalseNeg() to calculate the %FN for the data set that now has an analyte with an increased concentrations.</p> <p>(6) If the #FPs for that analyte has been reduced to 0, or the %FN for the data set exceeds the target value, the analyte concentration is returned to its previous value, the increment size is reduced, and the process in step(5) is repeated with the smaller increment instead of the initial increment.</p>

RunProgram() Macro Code	Function
	<p>(7) This process repeats until the increment has been reduced to some acceptably small value and either #FPs = 0 for each analyte or the concentrations of those with #FPs > 0 cannot be increased without exceeding the target %FN.</p> <p>(8) The size of the “acceptably small” increment is controlled by the value that you enter for Percent Precision on the [ControlScreen] page. See the instructions for FPMCalc on page 29 for more information about Percent Precision.</p> <p>(9) When the calculations have been completed for a specific %FN, the program compares the final analyte concentrations attained from raising them one increment at a time to the real analyte concentrations in the data set. It then assigns the real concentration that is closest to without exceeding the one found using increments.</p>
<p>If NumSteps > 1 Then Call FloatingPercentilesPass2</p> <p><u>Note:</u> FloatingPercentilesPass2() includes a large amount of explanatory text that is not included in this table. For additional information please see Error! Reference source not found. on page Error! Bookmark not defined..</p>	<p>If you are testing for more than one target %FN, the program carries out a second series of floating percentile calculations by calling the macro FloatingPercentilesPass2() in mod08FloatingPercentilesPass2, which does the following:</p> <p>(1) Selects the lowest result for each analyte from Pass1 and stores it in the row of "First Pass Results."</p> <p>(2) Clears all needed tables and recalculates the initial increment for each analyte.</p> <p>(3) Pastes the “First Pass Results” into the row for the first target %FN.</p> <p>(4) Repeats the process used in Pass1 for the first target %FN – find analyte with most #FPs; raise its concentration by one increment; recount #FPs and recalculate %FN; when necessary, go back one increment, make the increment smaller and try again until reaching an acceptably small concentration.</p> <p>(5) Unlike Pass1, Pass2 uses the final concentrations from the first target %FN as the initial concentrations for the second %FN, uses the final concentrations from the second target %FN as the initial concentrations for the third %FN, etc., until all of the target %FNs have been calculated.</p>
Application.StatusBar = False	Deletes the message from the status bar.
Call RecalcPerformance	Calls RecalcPerformance() in mod1RunProgram, which calculates the reliability measures for the final FPM values assigned by FloatingPercentilesPass2 (or by FloatingPercentilesPass1 if only one target %False Negative was evaluated).
Call DataFormat	Calls DataFormat() in mod10FormatPages, which centers the text and adjusts the column widths on the [Criteria] and [DataStorage] pages.
Time = (Timer - Temp) / 60	Calculates the program run time and saves it in minutes.
For i = 7 To 1 Step -1 Sheets(i).Activate	Restores each worksheet to the standard top-left position

RunProgram() Macro Code	Function
ActiveWindow.ScrollColumn = 1 ActiveWindow.ScrollRow = 1 Range("A1").Select Next i	
Sheets("Criteria").Select	Selects the page that will appear when the program ends.
Application.ScreenUpdating = True	Restores screenupdating so changes made during the running of the program will now show up on screen.
MsgBox (" Floating Percentile Calculations Completed ")	Puts message on screen to inform the user that the program has ended.

3.5.2 mod01RunProgram

This module contains 9 macros:

RunProgram controls the overall program by calling the macros that carry out the necessary sorting, testing, and calculating routines.

SetRanges initializes the variables by setting them equal to some of the relevant previously-declared ranges.

RankChem assigns a number to each analyte based on its rank from most (#1) to least false positives.

AETScreen tests each AET to see if it can be used in place of the regular FPM result. This is an option.

InputCheck checks to see if all of the required answers have been provided and if reasonable values have been entered in the "Program Settings" table on the [ControlScreen] page.

ClearTables deletes everything from the tables that store results so that data from a previous calculation do not interfere.

ClearTestTable deletes all data from the test table on the [Criteria] page and resets the initial counts to 0.

ClearAllTables deletes everything from the tables that store results as well as the analytical and bio data.

RecalcPerformance recalculates all of the performance measures when the FPM calculations have been completed.

The code and comments for this module are shown below.

```
' File: FPMCalc.xls

' This is the third of three spreadsheets used for implementing the
' Floating Percentile Method (FPM) developed by Teresa Michelsen of
' Avocet Consulting. The FPM is used to assess sediment toxicity
' and chemical composition data to develop sediment quality guidelines.

' FPMCalc.xls:
' (1) Accepts a table of chem and bio data created in FPMAnova.xls;
' (2) Sorts data by analyte into distributions from lowest to highest concentrations;
' (3) Uses the data distributions to generate data distribution percentiles;
' (4) Selects an initial FPM dataset based on user-entered targets of %False Negatives;
' (5) Compares the selected FPM dataset to the chem and bio data and counts the number of
' false positives and false negatives; and
' (5) Modifies the dataset in order to maintain the %False Negatives while minimizing the
' %False Positives.
```

```
' This spreadsheet was developed by Michael R. Anderson,
' Oregon Department of Environmental Quality.

' Last edited: December 6, 2008

' Names shown in [Square Brackets] in the comments are names of worksheets.
```

```
Option Private Module
Option Explicit
```

```
Public AETNote As Range, AETs As Range, AnovaFileName As Range
Public ChemBioData As Range, CriteriaHeader As Range
Public CriteriaSumDataRows As Range, CriteriaTable As Range
Public DataSummary As Range, DataSumName As Range
Public DataTable As Range, DistTable As Range, DistHeader As Range
Public DTHeader As Range, DTHit As Range, DTSample As Range
Public DTStation As Range, DTSurvey As Range, DTUnits As Range
Public Efficiency As Range, ErrorPercentiles As Range
Public ErrorTable As Range, EstimatedIterations As Range
Public FalseNegTargetHeader As Range, FalseNegTargetTable As Range
Public FalseHits As Range, FalseNoHits As Range
Public FalsePredHits As Range, FalsePredNoHits As Range, FirstPass As Range
Public FNInitial As Range, FNInterval As Range, FNFinal As Range
Public FullCriteriaTable As Range, FullDataTable As Range
Public IDHeader As Range, IDTable As Range, Increment As Range
Public IterationCount As Range, MaxConc As Range, MinConc As Range
Public NominalValues As Range, NumAETChem As Range
Public NumChem As Range, NumData As Range, NumDataHits As Range
Public NumDataIndets As Range, NumDataNoHits As Range, NumDataPoints As Range
Public NumFPMChem As Range, NumHits As Range, NumIncrements As Range
Public NumIndet As Range, NumNoHits As Range, NumOutliers As Range
Public NumSamples As Range, NumSteps As Range
Public OmitOutliers As Range, OutlierMultiple As Range
Public PercentFalseHits As Range, PercentFalseNegatives As Range
Public PercentFalseNoHits As Range, PercentFalsePositives As Range
Public PercentileHeader As Range, Percentiles As Range, PercentileTable As Range
Public PerformanceStats As Range, Precision As Range, PredHitSensitivity As Range
Public PredNoHitEfficiency As Range, Reliability As Range, ScreenAETs As Range
Public Sensitivity As Range, SortOrder As Range, SortSum As Range
Public StorageHeader As Range, TestCount As Range, TestCountStatus As Range
Public TestMass As Range, Time As Range, ToolBarStorage As Range
Public TrueHits As Range, TrueNoHits As Range, WatchCalc As Range, WorkRange As Range
Public TestIDTable As Range, TestDataTable As Range, TestTableHeader As Range
Public TestTrueHits As Range, TestTrueNoHits As Range
Public TestFalseHits As Range, TestFalseNoHits As Range
Public TestHitNoHit As Range, TestSurvey As Range, TestStation As Range
Public TestSample As Range, TestCorrect As Range
Public TestFalseNeg As Range, TestFalsePos As Range, VersionName As Range

Public i As Integer, j As Integer, k As Integer, m As Integer, n As Integer
Public Chem As Integer, CriteriaRow As Integer, Delta As Integer
Public IterationsRequested As Integer, FalseNegTarget As Integer
Public NewFalseNoHits As Integer, NumBioSets As Integer
Public PercentileRow As Integer, PredictedHitFlag As Integer
Public PredictedHits As Integer, PredictedNoHitFlag As Integer
Public SortNum As Integer

Public CommentYN As String, DestinationColumn As String, DestinationFile As String
Public SourceFile As String, ThisFileName As String

Public NewPercentFalseNegatives As Double
Public MaxHit As Double, MaxNoHit As Double, Temp As Double
```

```
' *****
' RunProgram() calls each main subroutine, adds notes to Status Bar to show
' progress of the calculations, and times the calculations.
' *****
```

```
Sub RunProgram()
```

```
Range("A1").Select
Call SetRanges
```

```
' If the program is revised change the version date below.
```

```
VersionName = "FPMCalc.xls Version 120608"
```

```
Time = ""
DataSummary.ClearContents
```

```
' Determine the number of false negative levels to calculate
' based on the requested initial, final, and interval values.
```

```
NumSteps = Int(1 + (FNFinal - FNInitial) / FNInterval)
```

```
Temp = Timer
```

```
Call Count
Application.ScreenUpdating = False
Call ClearTables
Call InputCheck
```

```
' Insert notes in the status bar so that user can follow
' the progress of the calculations then call relevant macros.
```

```
Application.StatusBar = "(1/6) Creating Distributions"
Call CreateDistributions
```

```
Application.StatusBar = "(2/6) Summarizing Data"
Call SummarizeData
```

```
Application.StatusBar = "(3/6) Calculating Percentiles"
Call CalculatePercentiles
```

```
Application.StatusBar = "(4/6) Calculating Error Levels"
Call ErrorCalculations
```

```
Application.StatusBar = "(5/6) Selecting Criteria"
Call SelectCriteria
```

```
Application.StatusBar = "(6/6) Calculating FloatingPercentiles"
Call FloatingPercentilesPass1
If NumSteps > 1 Then Call FloatingPercentilesPass2
```

```
Application.StatusBar = False
```

```
Call RecalcPerformance
Call DataFormat
```

```
Time = (Timer - Temp) / 60
```

```
' Restore each worksheet to the standard top-left position
```

```
For i = 7 To 1 Step -1
```

```

        Sheets(i).Activate
        ActiveWindow.ScrollColumn = 1
        ActiveWindow.ScrollRow = 1
        Range("A1").Select
    Next i

    Sheets("Criteria").Select

    Application.ScreenUpdating = True

    If CommentYN = "N" Then
        MsgBox ("    Floating Percentile Calculations Completed    ")
    Else
        MsgBox _
        ("    Floating Percentile Calculations Completed    " & vbCrLf & _
        "" & vbCrLf & _
        "Your lowest selected false negative target of " & FNInitial & "%" & vbCrLf & _
        "    was not achievable with this data set;" & vbCrLf & _
        "    " & NominalValues(3) & "%" & " has been substituted instead.")
    End If

End Sub

' *****
' SetRanges() initiates the variables by associating
' them with previously-defined ranges.
' *****

Sub SetRanges()

    Set AETs = Range("AETs")
    Set AETNote = Range("AETNote")
    Set AnovaFileName = Range("AnovaFileName")
    Set ChemBioData = Range("ChemBioData")
    Set CriteriaHeader = Range("CriteriaHeader")
    Set CriteriaTable = Range("CriteriaTable")
    Set DataSummary = Range("DataSummary")
    Set DataTable = Range("DataTable")
    Set DistHeader = Range("DistHeader")
    Set DistTable = Range("DistTable")
    Set DTHeader = Range("DTHeader")
    Set DTHit = Range("DTHit")
    Set DTSample = Range("DTSample")
    Set DTStation = Range("DTStation")
    Set DTSurvey = Range("DTSurvey")
    Set DTUnits = Range("DTUnits")
    Set Efficiency = Range("Efficiency")
    Set ErrorPercentiles = Range("ErrorPercentiles")
    Set ErrorTable = Range("ErrorTable")
    Set EstimatedIterations = Range("EstimatedIterations")
    Set FalseNoHits = Range("FalseNoHits")
    Set FalseHits = Range("FalseHits")
    Set FalsePredHits = Range("FalsePredHits")
    Set FalsePredNoHits = Range("FalsePredNoHits")
    Set FalseNegTargetHeader = Range("FalseNegTargetHeader")
    Set FalseNegTargetTable = Range("FalseNegTargetTable")
    Set FirstPass = Range("FirstPass")
    Set FNFinal = Range("FNFinal")
    Set FNInitial = Range("FNInitial")
    Set FNInterval = Range("FNInterval")
    Set FullCriteriaTable = Range("FullCriteriaTable")
    Set FullDataTable = Range("FullDataTable")

```

```

Set IDHeader = Range("IDHeader")
Set IDTable = Range("IDTable")
Set Increment = Range("Increment")
Set IterationCount = Range("IterationCount")
Set MaxConc = Range("MaxConc")
Set MinConc = Range("MinConc")
Set NumAETChem = Range("NumAETChem")
Set NumChem = Range("NumChem")
Set NumData = Range("NumData")
Set NumDataHits = Range("NumDataHits")
Set NumDataIndets = Range("NumDataIndets")
Set NumDataNoHits = Range("NumDataNoHits")
Set NumDataPoints = Range("NumDataPoints")
Set NumHits = Range("NumHits")
Set NumFPMChem = Range("NumFPMChem")
Set NumIncrements = Range("NumIncrements")
Set NumIndet = Range("NumIndet")
Set NumNoHits = Range("NumNoHits")
Set NumOutliers = Range("NumOutliers")
Set NumSamples = Range("NumSamples")
Set NumSteps = Range("NumSteps")
Set NominalValues = Range("NominalValues")
Set OmitOutliers = Range("OmitOutliers")
Set OutlierMultiple = Range("OutlierMultiple")
Set PercentFalseNegatives = Range("PercentFalseNegatives")
Set PercentFalsePositives = Range("PercentFalsePositives")
Set PercentileHeader = Range("PercentileHeader")
Set PercentileTable = Range("PercentileTable")
Set Percentiles = Range("Percentiles")
Set PerformanceStats = Range("PerformanceStats")
Set Precision = Range("Precision")
Set PredHitSensitivity = Range("PredHitSensitivity")
Set PredNoHitEfficiency = Range("PredNoHitEfficiency")
Set Reliability = Range("Reliability")
Set ScreenAETs = Range("ScreenAETs")
Set Sensitivity = Range("Sensitivity")
Set SortOrder = Range("SortOrder")
Set SortSum = Range("SortSum")
Set StorageHeader = Range("StorageHeader")
Set TestCountStatus = Range("TestCountStatus")
Set TestMass = Range("TestMass")
Set TestDataTable = Range("TestDataTable")
Set TestIDTable = Range("TestIDTable")
Set TestTableHeader = Range("TestTableHeader")
Set TestTrueHits = Range("TestTrueHits"): Set TestTrueNoHits = Range("TestTrueNoHits")
Set TestFalseHits = Range("TestFalseHits"): Set TestFalseNoHits = Range("TestFalseNoHits")
Set TestCount = Range("TestCount")
Set TestHitNoHit = Range("TestHitNoHit"): Set TestSurvey = Range("TestSurvey")
Set TestSample = Range("TestSample"): Set TestStation = Range("TestStation")
Set TestCorrect = Range("TestCorrect"): Set TestFalseNeg = Range("TestFalseNeg")
Set TestFalsePos = Range("TestFalsePos")
Set Time = Range("Time")
Set ToolBarStorage = Range("ToolBarStorage")
Set TrueHits = Range("TrueHits")
Set TrueNoHits = Range("TrueNoHits")
Set VersionName = Range("VersionName")
Set WatchCalc = Range("WatchCalc")

End Sub

```



```

' *****
' RankChem() assigns a number to each analyte based on its rank
' from most (#1) to least false positives. It ignores any analyte
' that already has been assigned a final concentration (i.e., any analyte
' that has been given one of the three symbols (+, -, or AET) during
' the calculations by the ScreenAETs or FloatingPercentiles subroutines.
' *****

Sub RankChem()

Dim Countout As Integer

' Sort the columns first by those that contain one of the three symbols
' shown above (the FP count has already been completed or is being skipped in
' these columns), determine how many of these columns there are, and sort
' the remaining columns (the ones for which the FP count will continue) by the
' current number of false positives (high to low), followed by the number of
' iterations that have already taken place (low to high), and finally by the
' current calculated screening value (low to high).

FullCriteriaTable.Sort _
    Key1:=FullCriteriaTable(CriteriaRow + 5, 1), Order1:=xlDescending, _
    Header:=xlNo, OrderCustom:=1, _
    Orientation:=xlLeftToRight

' Count the number of cells that have data in them. These analytes have already
' been evaluated and are marked as "+", "-", or "AET"). The sort routine below only
' needs to sort the analytes that are still being evaluated, therefore the column
' where the next iteration starts is located at Countout + 1.

Countout = Application.CountA(Range(FullCriteriaTable(CriteriaRow + 5, 1),
FullCriteriaTable(CriteriaRow + 5, NumChem)))

Range(FullCriteriaTable(1, Countout + 1), FullCriteriaTable(65, NumChem)).Sort _
    Key1:=FullCriteriaTable(CriteriaRow + 4, 1), Order1:=xlDescending, _
    Key2:=IterationCount, Order2:=xlAscending, _
    Key3:=FullCriteriaTable(CriteriaRow + 3, 1), Order3:=xlAscending, _
    Header:=xlNo, OrderCustom:=1, _
    Orientation:=xlLeftToRight

SortNum = 1

For n = 1 To NumChem
    If FullCriteriaTable(CriteriaRow + 5, n) = "" Then
        SortOrder(n) = SortNum
        SortNum = SortNum + 1
    End If
Next n

' Put the columns back into alphabetical order to be
' consistent with the column order for the other worksheets

FullCriteriaTable.Sort _
    Key1:=FullCriteriaTable(1, 1), Order1:=xlAscending, _
    Header:=xlNo, OrderCustom:=1, _
    Orientation:=xlLeftToRight

End Sub

```

```

' *****
' AETScreen() is an option that you can select by answering "Y"
' to the "Pre-Screen AETs (Y or N)?" question on the [ControlScreen]
' page. If selected, this macro tests each AET to see if it can be
' used in place of the regular FPM result.
' *****

Sub AETScreen()

For i = 1 To NumChem

' Place the AET concentration into the cell that stores the FPM result
' (CriteriaRow + 2, i). Then count the false negatives from all stations.
' If %FN does not increase, add a note to indicate that the AET has been
' used. If %FN increases, restore the initial concentration and retain
' for full FPM calculation.

CriteriaTable(CriteriaRow + 2, i) = AETs(i)
Call CountFalseNeg

If NewPercentFalseNegatives = FalseNegTargetTable(CriteriaRow, 1) Then
CriteriaTable(CriteriaRow + 3, i) = 0
CriteriaTable(CriteriaRow + 4, i) = "AET"
Else
CriteriaTable(CriteriaRow + 2, i) = CriteriaTable(CriteriaRow, i)
End If

Next i

End Sub

' *****
' InputCheck() checks to see if all of the required Yes/No answers
' have been provided and if reasonable values for the range of
' %False Negatives have been entered in the "Program Settings"
' table on the [ControlScreen] page.
' *****

Sub InputCheck()

ScreenAETs = UCase(ScreenAETs)
If (ScreenAETs <> "Y" And ScreenAETs <> "N") Then
MsgBox ("Pre-Screen AETs?" must be either Y or N")
ScreenAETs.Select
Application.StatusBar = False
End
End If

OmitOutliers = UCase(OmitOutliers)
If (OmitOutliers <> "Y" And OmitOutliers <> "N") Then
MsgBox ("Omit Outliers?" must be either Y or N")
OmitOutliers.Select
Application.StatusBar = False
End
End If

WatchCalc = UCase(WatchCalc)
If (WatchCalc <> "Y" And WatchCalc <> "N") Then
MsgBox ("Watch Calculations?" must be either Y or N")
WatchCalc.Select

```

```

        Application.StatusBar = False
    End
End If

If FNFinal < FNInitial Then
    MsgBox ("Final False Neg Target must be greater than or equal to Initial value.")
    FNFinal.Select
    Application.StatusBar = False
End
End If

If NumSteps > 10 Then
    Sheets("ControlScreen").Select

    MsgBox ( _
        "
        You cannot request more than 10 steps." & Chr(13) & _
        "
        Modify the entries in the 'Criteria Table Settings' box." & Chr(13)
    & _
        "" & Chr(13) & _
        "No. Steps = 1 + (Final False Neg Target - Initial False Neg Target)/Target Interval")

    Application.StatusBar = False
End
End If

End Sub

' *****
' ClearTables() deletes everything from the tables that store results
' so that data from a previous calculation do not interfere.
' *****

Sub ClearTables()

Call SetRanges

DistHeader.ClearContents
DistTable.ClearContents
ErrorTable.ClearContents
FalseNegTargetTable.ClearContents
FirstPass.ClearContents
FullCriteriaTable.ClearContents
NominalValues.ClearContents
PercentileHeader.ClearContents
PercentileTable.ClearContents

End Sub

' *****
' ClearTestTable() deletes all data from the test table
' on the [Criteria] page and resets the initial counts to 0.
' *****

Sub ClearTestTable()

Call SetRanges

Application.ScreenUpdating = False

TestIDTable.ClearContents
TestDataTable.ClearContents

```

```

TestTableHeader.ClearContents
TestCount.ClearContents
TestMass.ClearContents
PerformanceStats.ClearContents

TestTrueHits = 0: TestTrueNoHits = 0
TestFalseHits = 0: TestFalseNoHits = 0

Application.ScreenUpdating = True

End Sub

' *****
' ClearAllTables() deletes everything from the tables that
' store results as well as the analytical and bio data.
' *****

Sub ClearAllTables()

Dim Response As String

Response = MsgBox("Are you sure you want to clear ALL tables, including the Table of
Biological and Analytical Data?", vbYesNoCancel)
If Response <> vbYes Then End
Call ClearTables
Call ClearTestTable
DataTable.ClearContents
DTHeader.ClearContents
DTUnits.ClearContents
IDTable.ClearContents

End Sub

' *****
' RecalcPerformance() recalculates all of the performance
' measures when the FPM calculations have been completed.
' *****

Sub RecalcPerformance()

If NumSteps > 1 Then
For CriteriaRow = 1 To (5 * NumSteps - 4) Step 5

    TestMass.ClearContents
    Range(CriteriaTable(CriteriaRow, 1), CriteriaTable(CriteriaRow, NumChem)).Copy
    Range(TestMass(1), TestMass(NumChem)).PasteSpecial Paste:=xlPasteValues

    Call CountTrueFalse

    For i = 1 To 7
        PerformanceStats(i).Copy
        FalseNegTargetTable(CriteriaRow, i).PasteSpecial Paste:=xlPasteValues
    Next i

Next CriteriaRow
End If

For CriteriaRow = 1 To (5 * NumSteps - 4) Step 5

    TestMass.ClearContents
    Range(CriteriaTable(CriteriaRow + 2, 1), CriteriaTable(CriteriaRow + 2, NumChem)).Copy

```

```
Range (TestMass (1), TestMass (NumChem)) .PasteSpecial Paste:=xlPasteValues

Call CountTrueFalse

For i = 1 To 7
    PerformanceStats(i).Copy
    FalseNegTargetTable(CriteriaRow + 2, i).PasteSpecial Paste:=xlPasteValues
Next i

Next CriteriaRow

Call ClearTestTable

End Sub
```

This is the end of FPMCalc.xls mod01RunProgram.

3.5.3 mod02CreateDistributions

This module contains only 1 macro, **CreateDistributions**. The code and comments are shown below.

```
Option Private Module
Option Explicit

' *****
' CreateDistributions() sorts the data from the [DataTable] page
' and lists it in columns from lowest concentration to highest
' concentration on the [Distributions] page.
' *****

Sub CreateDistributions()

Sheets("Distributions").Select

' Enter chemical names from DataTable into the
' header rows of the DistributionTable.

For i = 1 To NumChem
    DistHeader(i) = DTHeader(i)
Next i

' Copy data from DataTable to DistributionTable

For j = 1 To NumChem
    For i = 1 To NumSamples
        DistTable(i, j) = DataTable(i, j)
    Next i
Next j

' Sort data in each column of DistributionTable from low to high

For i = 1 To NumChem
    Range(DistTable(1, i), DistTable(NumSamples, i)).Select
    Selection.Sort _
        Key1:=DistTable(1, i), Order1:=xlAscending, _
        Header:=xlNo, Orientation:=xlTopToBottom
Next i

DistHeader.Select
Call CenterText
Range("A1").Select

End Sub
```

This is the end of FPMCalc.xls mod02CreateDistributions.

3.5.4 mod03SummarizeData

This module contains only 1 macro, SummarizeData . The code and comments are shown below.
--

```
Option Private Module
Option Explicit

' *****
' SummarizeData() summarizes key parameters for each chemical and
' stores them on the [Criteria] page. It also assigns an AET to
' each chemical and provides an option to screen out outliers
' when assigning the AET.
' *****

Sub SummarizeData()

' Sort the Hit/No-Hit entries in the Data Table from low to high,
' which orders them as indeterminates (-1), no-hits (0), and hits(1).

Sheets("DataTable").Select
ChemBioData.Select

Selection.Sort Key1:=ChemBioData(1, 1), Order1:=xlAscending, _
    Header:=xlNo, OrderCustom:=1, Orientation:=xlTopToBottom _

' Count number of data points, find the maximum, minimum, etc. for each analyte.
For i = 1 To NumChem

    Range(DataTable(1, i), DataTable(NumSamples, i)).Select
    Set WorkRange = Selection
    NumDataPoints(i) = Application.CountA(WorkRange)

    MinConc(i) = Application.Min(WorkRange)
    MaxConc(i) = Application.Max(WorkRange)

' Select the Hits, count them, and find the max.

    Range(DataTable(NumIndet + NumNoHits + 1, i), DataTable(NumSamples, i)).Select
    Set WorkRange = Selection
    NumDataHits(i) = Application.CountA(WorkRange)
    MaxHit = Application.Max(WorkRange)

' Select the No-Hits, count them, and find the max.

    Range(DataTable(NumIndet + 1, i), DataTable(NumIndet + NumNoHits, i)).Select
    Set WorkRange = Selection
    NumDataNoHits(i) = Application.CountA(WorkRange)
    MaxNoHit = Application.Max(WorkRange)

    NumDataIndets(i) = NumDataPoints(i) - NumDataHits(i) - NumDataNoHits(i)

' Assign an AET to the chemical
```

```

' If there are no "No Hits" then an AET cannot be identified.
' This is marked "-" and a note is placed in the AETNote row.

If NumDataNoHits(i) = 0 Then
    AETs(i) = "-"
    AETNote(i) = "No NH"
End If

' If OmitOutliers = "Y" then outliers will not be considered in the
' selection of an AET. The highest value is considered an outlier if it
' is greater than the next highest value by a specified number of times.
' If that is the case, then the 2nd highest is compared to the 3rd highest,
' etc. until the highest No Hit is found that passes the outlier test.
' If the AETs fail the outlier test until there is only one No Hit
' remaining, then AET = the last (smallest) No Hit. There must be 2
' or more No Hits to apply this test.

If (OmitOutliers = "Y" And AETs(i) <> "-" And NumDataNoHits(i) > 1) Then

    For k = 1 To (NumDataNoHits(i) - 1)
        If Application.Large(WorkRange, k) <= OutlierMultiple *
Application.Large(WorkRange, k + 1) Then
            AETs(i) = Application.Large(WorkRange, k)
            Exit For
        ElseIf k = (NumDataNoHits(i) - 1) Then
            AETs(i) = Application.Large(WorkRange, k + 1)
            Exit For
        End If
    Next k
    NumOutliers(i) = k - 1
    If AETs(i) >= MaxHit Then AETNote(i) = ">"
Else
    AETs(i) = MaxNoHit
End If

Next i

End Sub

```

This is the end of FPMCalc.xls mod03SummarizeData.
--

3.5.5 mod04CalculatePercentiles

This module contains only 1 macro, **CalculatePercentiles**. The code and comments are shown below.

```
Option Private Module
Option Explicit

' *****
' CalculatePercentiles() creates a percent distribution list
' for the concentrations of each analyte based on the raw data
' distributions on the [Distributions] page. The results are
' stored on the [Percentiles] page.
' *****

Sub CalculatePercentiles()

Sheets("Percentiles").Select

' Copy chemical names into the header of the percentile table.

For i = 1 To NumChem
    PercentileHeader(i) = DTHeader(i)
Next i

PercentileHeader.Select
Call CenterText
Range("A1").Select

Sheets("Distributions").Select

' Calculate percentiles of the range of data for each chemical

For i = 1 To NumChem
    If OmitOutliers = "Y" Then
        Range(DistTable(1, i), DistTable(NumDataPoints(i) - NumOutliers(i), i)).Select
    Else
        Range(DistTable(1, i), DistTable(NumDataPoints(i), i)).Select
    End If

    Set WorkRange = Selection
    For j = 1 To 100
        PercentileTable(j, i) = Application.Percentile(WorkRange, j / 100)
    Next j
Next i

Sheets("Percentiles").Select
Call CenterText

End Sub
```

This is the end of FPMCalc.xls mod04CalculatePercentiles.

3.5.6 mod05ErrorCalculations

This module contains only 1 macro, **ErrorCalculations**. The code and comments are shown below.

```
Option Private Module
Option Explicit
```

```
' *****
' ErrorCalculations() compares each row of percentile concentrations to
' station chemical concentrations and determines if the percentile concentrations
' would correctly predict each hit and no-hit, and counts the number of
' true/false hits/no-hits for each row.
' *****

Sub ErrorCalculations()

Sheets("ErrorCalc").Select

For i = 1 To 100 'Percentile Rows

TrueHits(i) = 0: TrueNoHits(i) = 0
FalseHits(i) = 0: FalseNoHits(i) = 0

    For j = 1 To NumSamples
        PredictedHitFlag = 0
        For k = 1 To NumChem
            If DataTable(j, k) <> "" Then
                If DataTable(j, k) > PercentileTable(i, k) Then
                    PredictedHitFlag = 1
                    Exit For
                End If
            End If
        Next k

        If (PredictedHitFlag = 1 And DTHit(j) = 1) Then
            TrueHits(i) = TrueHits(i) + 1
        ElseIf (PredictedHitFlag = 0 And DTHit(j) = 0) Then
            TrueNoHits(i) = TrueNoHits(i) + 1
        ElseIf (PredictedHitFlag = 1 And DTHit(j) = 0) Then
            FalseHits(i) = FalseHits(i) + 1
        ElseIf (PredictedHitFlag = 0 And DTHit(j) = 1) Then
            FalseNoHits(i) = FalseNoHits(i) + 1
        End If

    Next j

Next i

For i = 1 To 100

' Calculate DOE evaluation parameters

PercentFalseNegatives(i) = 100 * FalseNoHits(i) / (TrueHits(i) + FalseNoHits(i))
PercentFalsePositives(i) = 100 * FalseHits(i) / (TrueNoHits(i) + FalseHits(i))
Sensitivity(i) = 100 * TrueHits(i) / (TrueHits(i) + FalseNoHits(i))
Efficiency(i) = 100 * TrueNoHits(i) / (TrueNoHits(i) + FalseHits(i))
Reliability(i) = 100 * (TrueHits(i) + TrueNoHits(i)) / (TrueHits(i) + TrueNoHits(i) +
FalseHits(i) + FalseNoHits(i))
```

```

' Calculate DEQ evaluation parameters

If (TrueHits(i) = 0 And FalseHits(i) = 0) Then
    FalsePredHits(i) = "NA"
    PredHitSensitivity(i) = "NA"
Else
    FalsePredHits(i) = 100 * FalseHits(i) / (TrueHits(i) + FalseHits(i))
    PredHitSensitivity(i) = 100 * TrueHits(i) / (TrueHits(i) + FalseHits(i))
End If

If (TrueNoHits(i) = 0 And FalseNoHits(i) = 0) Then
    FalsePredNoHits(i) = "NA"
    PredNoHitEfficiency(i) = "NA"
Else
    FalsePredNoHits(i) = 100 * FalseNoHits(i) / (TrueNoHits(i) + FalseNoHits(i))
    PredNoHitEfficiency(i) = 100 * TrueNoHits(i) / (TrueNoHits(i) + FalseNoHits(i))
End If

Next i

End Sub

```

This is the end of FPMCalc.xls mod05ErrorCalculations.
--

3.5.7 mod06SelectCriteria

This module contains only 1 macro, **Select Criteria**. The code and comments are shown below.

```
Option Private Module
Option Explicit

' *****
' SelectCriteria() finds the data percentiles closest to the specified percent
' false negatives and copies those sets of percentiles to the [Criteria] page.
' *****

Sub SelectCriteria()

Dim Start As Integer, Response As String
Dim ErrorHandler
Set ErrorHandler = Range("ErrorHandler")

' This enters the data table row names on the "Criteria" worksheet
Sheets("Criteria").Select

' Print names in the header row on the Criteria page

For i = 1 To NumChem: CriteriaHeader(i) = PercentileHeader(i): Next i

CriteriaRow = 1
CommentYN = "N"

For FalseNegTarget = FNInitial To FNFinal Step FNInterval
    NominalValues(CriteriaRow + 2) = FalseNegTarget

    For PercentileRow = 1 To 100
        If PercentFalseNegatives(PercentileRow) > FalseNegTarget Then

' If the the lowest false negative percentage of the data set (located in
' PercentileRow = 1) is greater than the Initial False Negative Target, the
' target percentage is reset to the next integral value above the lowest value
' and the calculation then proceeds. The remaining False Negative Targets are
' not changed.

            If PercentileRow = 1 Then
                NominalValues(CriteriaRow + 2) = Int(PercentFalseNegatives(1)) + 1
                Start = 0
                CommentYN = "Y"
            Else
                Start = 1
            End If

            Exit For
        End If
    Next PercentileRow

    For i = 1 To 6
        FalseNegTargetTable(CriteriaRow, i) = ErrorTable(PercentileRow - Start, i + 4)
    Next i
```

```
FalseNegTargetTable(CriteriaRow, 7) = ErrorTable(PercentileRow - Start, 13)

For i = 1 To NumChem
    CriteriaTable(CriteriaRow, i) = PercentileTable(PercentileRow - Start, i)
Next i

CriteriaRow = CriteriaRow + 5
Next FalseNegTarget

Range("A1").Select

End Sub
```

This is the end of FPMCalc.xls mod06SelectCriteria.

3.5.8 mod07FloatingPercentilePass1

This module contains only 1 macro, **FloatingPercentilesPass1**. The code and comments are shown below.

```
Option Private Module
Option Explicit
```

```
' *****
' FloatingPercentilesPass1() finds an initial set of FPM results starting
' with initial values taken from the [Percentiles] page.
' *****

Sub FloatingPercentilesPass1()

' Data stored under the chemical names on the CriteriaRow page are displayed
' in sets of 5 rows as follows:

' CriteriaRow holds the initial concentrations determined by the allowable %FN
' (CriteriaRow + 1) holds the #FP for each initial concentration in CriteriaRow
' (CriteriaRow + 2) holds the final concentrations determined by this method
' (CriteriaRow + 3) holds the #FP for the final concentrations in (CriteriaRow + 2)
' (CriteriaRow + 4) holds the code indicating which of the two limiting
' criteria controls each final concentration in (CriteriaRow + 2)

' (Note: %FN = the percentage of false negative results for a given data set, and
' #FP = the number of false positives for a given analyte in the latest round of tests.)

' The program starts the Floating Percentile calculations with the analyte that
' has the highest #FPs. Its concentration is increased by a specified increment
' and the %FN and #FPs are then reevaluated. When it is determined that either
' the %FN > the desired %FN or the #FPs = 0, the size of the increment for that
' specific analyte is reduced and the entire process is started again. Each time
' the increment is reduced it is considered another "iteration" for that analyte.
' So, each analyte starts out on iteration #1 with the initial increment and the
' number of iterations is increased every time that the increment is reduced.

' After each test the program recounts the #FPs and continues to use the analyte
' with the most #FPs. If two or more analytes are tied for the highest #FPs, the
' program uses the analyte with the fewest number of iterations. If two or more
' analytes are tied for fewest nubmer of iterations, the program uses the analyte
' that currently has the lowest assigned screening level.

' Use the minimum and maximum concentrations of each analyte to evaluate the
' initial increments for calculating the number of false positive values. Then
' estimate the number of iterations necessary to reach the desired percent precision.

For i = 1 To NumChem
    Increment(i) = (MaxConc(i) - MinConc(i)) / NumIncrements
    If Increment(i) > 0 Then
        EstimatedIterations(i) = Int(Application.Log10(100 * (MaxConc(i) - MinConc(i)) /
(Precision * MinConc(i))) / Application.Log10(NumIncrements))
    Else
        EstimatedIterations(i) = "NA"
    End If
Next i
```

```

For CriteriaRow = 1 To (5 * NumSteps - 4) Step 5

'   This copies the initial concentration of each analyte (for the specified %FN)
'   into the "final concentration" row prior to starting the calculations. All changes
'   in that concentration will be made in that row until the final concentration is
'   reached, which is based either on #FPs = 0 or %FN will increase if the concentration
'   is increased any higher. This also sets the initial iteration counts to 0.

For i = 1 To NumChem
    CriteriaTable(CriteriaRow + 2, i) = CriteriaTable(CriteriaRow, i)
    IterationCount(i) = 1
Next i

'   Before starting the Floating Percentile calculations, you have the option of
'   screening out analytes which have AET concentrations that do not increase the %FN
'   above the acceptable level. If you use this option, analytes that behave this way
'   are designated "AET" in the symbols row and these analytes are not included in the
'   subsequent Floating Percentile calculations. This can significantly speed up the
'   calculations since there are cases where a large number of analytes have AET values
'   that fit this criterion. Note, however, that the results will not necessarily be
'   identical to those in which all analytes go through the full Floating Percentile
'   calculations

If ScreenAETs = "Y" Then Call AETScreen

Do

'   Set the sorting counter to zero and clear all entries from the row
'   that designates the order in which the analytes should be sorted.

SortSum = 0
For Chem = 1 To NumChem: SortOrder(Chem) = "": Next Chem

'   Count #FPs for each chemical, then sort the data in decreasing
'   order of #FPs, assign a number to each chemical (#1= most #FPs),
'   and write this number in the SortOrder row.

Call CountFalsePos(2)
Call RankChem

'   Select the chemical with the highest #FPs, which
'   is the one that was assigned "1" in the previous sort.
'   Use that chemical for the subsequent tests within this loop.

For Chem = 1 To NumChem
    If SortOrder(Chem) = 1 Then Exit For
Next Chem

'   Count the number of chemicals that are still being evaluated
'   (i.e., those with numbers in the 'SortOrder' data row).

SortSum = Application.WorksheetFunction.Count(SortOrder)
Application.StatusBar = "(6/6) Calculating FloatingPercentiles (1st Pass): " &
NominalValues(CriteriaRow + 2) & "% False Negatives - " & SortSum & " Chemicals Left to
Evaluate"

If SortSum = 0 Then Exit Do

```

```

'   Raise the value of the selected chemical (i.e., increase CurrentDataIndex(Chem))
'   and see if it meets one of the limiting criteria listed below, which then causes an exit.
'       If the #FP = 0
'       If the %FN > the designated level

CriteriaTable(CriteriaRow + 2, Chem) = CriteriaTable(CriteriaRow + 2, Chem) + Increment(Chem)

'   Recount the number of false positives for each chemical at each station and
'   count total false negatives for each data set (i.e., total from all stations).

Call CountFalsePos(2)
Call CountFalseNeg

'   This sets the number of iterations to use for each calculation
'   based on number calculated from the desired percent percision.

IterationsRequested = EstimatedIterations(Chem)

'   If percent false negatives (%FN) exceeds specified value, go back one increment,
'   set the increment to a smaller value and try again. Assign the symbol "-" to the
chemical
'   after the specified number of iterations has been reached.

If NewPercentFalseNegatives > NominalValues(CriteriaRow + 2) Then
    CriteriaTable(CriteriaRow + 2, Chem) = CriteriaTable(CriteriaRow + 2, Chem) - Increment(Chem)

    If IterationCount(Chem) + 1 > IterationsRequested Then
        CriteriaTable(CriteriaRow + 4, Chem) = "-"
    Else
        IterationCount(Chem) = IterationCount(Chem) + 1
        Increment(Chem) = Increment(Chem) / NumIncrements
    End If

'   If number of false positives (#FPs) has been reduced to zero, go back one increment,
'   set the increment to a smaller value and try again. Assign the symbol "+" to the chemical
'   and exit the loop after the specified number of iterations.

ElseIf CriteriaTable(CriteriaRow + 3, Chem) = 0 Then
    If IterationCount(Chem) + 1 > IterationsRequested Then
        CriteriaTable(CriteriaRow + 4, Chem) = "+"
    Else
        CriteriaTable(CriteriaRow + 2, Chem) = CriteriaTable(CriteriaRow + 2, Chem) - Increment(Chem)
        Increment(Chem) = Increment(Chem) / NumIncrements
        IterationCount(Chem) = IterationCount(Chem) + 1
    End If

'   If the %FN is still acceptable and the #FPs has not been reduced to zero but the
'   tested concentration exceeds the maximum concentration (very unlikely unless a large
'   number of iterations or a very small precision is requested), set the increment to a
'   smaller value and try again. Assign the symbol "max" to the chemical and exit the
'   loop after the specified number of iterations.

ElseIf CriteriaTable(CriteriaRow + 2, Chem) > MaxConc(Chem) Then
    If IterationCount(Chem) + 1 > IterationsRequested Then
        CriteriaTable(CriteriaRow + 4, Chem) = "max"
    Else
        CriteriaTable(CriteriaRow + 2, Chem) = CriteriaTable(CriteriaRow + 2, Chem) - Increment(Chem)
        Increment(Chem) = Increment(Chem) / NumIncrements
        IterationCount(Chem) = IterationCount(Chem) + 1
    End If

End If

```



```

If WatchCalc = "Y" Then
    Application.ScreenUpdating = True
    Application.ScreenUpdating = False
End If

Loop

' This determines which real data points are closest to but do not
' exceed the values assigned by the Floating Percentile calculations.
' The real data points are used for the final results.

For i = 1 To NumChem
    For j = 1 To NumDataPoints(i)
        If DistTable(j, i) > CriteriaTable(CriteriaRow + 2, i) Then
            CriteriaTable(CriteriaRow + 2, i) = DistTable(j - 1, i)
            Exit For
        ElseIf j = NumDataPoints(i) Then
            CriteriaTable(CriteriaRow + 2, i) = DistTable(j, i)
            Exit For
        End If
    Next j
Next i

' Reset all increments and iteration counts before
' starting to evaluate the next row.

For i = 1 To NumChem
    Increment(i) = (MaxConc(i) - MinConc(i)) / NumIncrements
Next i

Next CriteriaRow

If NumSteps = 1 Then Call CountAETs

End Sub

```

This is the end of FPMCalc.xls mod07FloatingPercentilePass1.
--

3.5.9 mod08FloatingPercentilePass2

This module contains only 1 macro, **FloatingPercentilesPass2**. The code and comments are shown below.

```
Option Private Module
Option Explicit
```

```
' *****
' FloatingPercentilesPass2() refines the FPM results by starting with the
' lowest initial FPM results and running them through the process a second time.
' *****

Sub FloatingPercentilesPass2()

' This spreadsheet can calculate results either for a single specified percent false negatives
' (%FNs) or for multiple numbers of %FNs up to a total of 10. (See "Criteria Table Settings"
' on the [ControlScreen] page.) When two or more results are requested the program uses this
' optimization step, which works as described in this example. Let's say that you want the
' spreadsheet to calculate the results for 5%, 10%, 15%, 20% & 25%. Those calculations are
' carried out in FloatingPercentilesPass1. After obtaining the results, this routine scans
' them to determine the lowest concentration for each analyte in its set of five results (5%FNs
' to 25%FNs). The lowest results are then placed into the first row of the Criteria Table
' (replacing the original set) and used as the starting point for a second calculation of the
' 5%FN results. This second set of results for 5%FNs are then used as the starting point for
' the second set of 10%FN calculations. The second set of results are used as the starting point
' for the next higher %FNs until completing the highest target, which is 25% in this example.

' Select the lowest result for each analyte and store it in row of "First Pass Results."

For i = 1 To NumChem
    FirstPass(i) = CriteriaTable(3, i)
    For j = 8 To (5 * NumSteps - 2) Step 5
        If CriteriaTable(j, i) < FirstPass(i) Then FirstPass(i) = CriteriaTable(j, i)
    Next j
Next i

' Clear the tables in preparation for the second round of calculations.

CriteriaTable.ClearContents
FalseNegTargetTable.ClearContents

For CriteriaRow = 1 To (5 * NumSteps - 4) Step 5

' If this is the first %FN target value to be recalculated, copy the First Pass Results
' into the "initial concentration" row, where they will remain unchanged, and into the
' "final concentration" row where they will be stepped up during the course of the
' calculations. If this not the first %FN target, then copy the final concentrations from
' the previous target value into the initial and final concentration rows. During the
' calculations all changes in concentrations will be made in the final concentration row
' until the actual final concentration is reached, which is based either on #FPs = 0 or
' %FN will increase if the concentration is increased any higher. This also resets the
' initial iteration counts to 1.
```

```

For i = 1 To NumChem
    If CriteriaRow = 1 Then
        CriteriaTable(1, i) = FirstPass(i)
        CriteriaTable(3, i) = FirstPass(i)
    Else
        CriteriaTable(CriteriaRow, i) = CriteriaTable(CriteriaRow - 3, i)
        CriteriaTable(CriteriaRow + 2, i) = CriteriaTable(CriteriaRow - 3, i)
    End If
    IterationCount(i) = 1
Next i

Call CountFalsePos(0)

Do

    ' Set all of the counters to zero

SortSum = 0
For Chem = 1 To NumChem: SortOrder(Chem) = "": Next Chem

    ' Count #FPs for each chemical, then sort the data in decreasing
    ' order of #FPs, assign a number to each chemical (#1= most #FPs),
    ' and write this number in the SortOrder row.

Call CountFalsePos(2)
Call RankChem

    ' Select the chemical with the highest #FPs, which
    ' is the one that was assigned "1" in the previous sort.
    ' Use that chemical for the subsequent tests within this loop.

For Chem = 1 To NumChem
    If SortOrder(Chem) = 1 Then Exit For
Next Chem

    ' Count the number of chemicals that are still being evaluated
    ' (i.e., those with numbers in the 'SortOrder' data row).

SortSum = Application.WorksheetFunction.Count(SortOrder)
Application.StatusBar = "(6/6) Calculating FloatingPercentiles (2nd Pass): " &
NominalValues(CriteriaRow + 2) & "% False Negatives - " & SortSum & " Chemicals Left to
Evaluate"
If SortSum = 0 Then Exit Do

    ' Raise the value of the selected chemical (i.e., increase CurrentDataIndex(Chem))
    ' and see if it meets one of the limiting criteria listed below, which then causes an exit.
    '     If the #FP = 0
    '     If the %FN > the designated level

CriteriaTable(CriteriaRow + 2, Chem) = CriteriaTable(CriteriaRow + 2, Chem) + Increment(Chem)

    ' Recount the number of false positives for each chemical at each station and
    ' count total false negatives for each data set (i.e., total from all stations).

Call CountFalsePos(2)
Call CountFalseNeg

```

```

' This sets the number of iterations to use for each calculation
' based on number calculated from the desired percent percision.

IterationsRequested = EstimatedIterations(Chem)

' If percent false negatives (%FN) exceeds specified value, go back one increment,
' set the increment to a smaller value and try again. Assign the symbol "-" to the chemical
' after the specified number of iterations has been reached.

If NewPercentFalseNegatives > NominalValues(CriteriaRow + 2) Then
    CriteriaTable(CriteriaRow + 2, Chem) = CriteriaTable(CriteriaRow + 2, Chem) -
Increment(Chem)

    If IterationCount(Chem) + 1 > IterationsRequested Then
        CriteriaTable(CriteriaRow + 4, Chem) = "-"
    Else
        IterationCount(Chem) = IterationCount(Chem) + 1
        Increment(Chem) = Increment(Chem) / NumIncrements
    End If

' If number of false positives (#FPs) has been reduced to zero, go back one increment,
' set the increment to a smaller value and try again. Assign the symbol "+" to the chemical
' and exit the loop after the specified number of iterations.

ElseIf CriteriaTable(CriteriaRow + 3, Chem) = 0 Then
    If IterationCount(Chem) + 1 > IterationsRequested Then
        CriteriaTable(CriteriaRow + 4, Chem) = "+"
    Else
        CriteriaTable(CriteriaRow + 2, Chem) = CriteriaTable(CriteriaRow + 2, Chem) - Increment(Chem)
        Increment(Chem) = Increment(Chem) / NumIncrements
        IterationCount(Chem) = IterationCount(Chem) + 1
    End If

' If the %FN is still acceptable and the #FPs has not been reduced to zero but the
' tested concentration exceeds the maximum concentration (very unlikely unless a large
' number of iterations or a very small precision is requested), set the increment to a
' smaller value and try again. Assign the symbol "max" to the chemical and exit the
' loop after the specified number of iterations.

ElseIf CriteriaTable(CriteriaRow + 2, Chem) > MaxConc(Chem) Then
    If IterationCount(Chem) + 1 > IterationsRequested Then
        CriteriaTable(CriteriaRow + 4, Chem) = "max"
    Else
        CriteriaTable(CriteriaRow + 2, Chem) = CriteriaTable(CriteriaRow + 2, Chem) - Increment(Chem)
        Increment(Chem) = Increment(Chem) / NumIncrements
        IterationCount(Chem) = IterationCount(Chem) + 1
    End If

End If

If WatchCalc = "Y" Then
    Application.ScreenUpdating = True
    Application.ScreenUpdating = False
End If

Loop

```

```

' This determines which real data point is closest to but does not
' exceed the value assigned by the Floating Percentile calculation.
' The real data point is used for the final result.

For i = 1 To NumChem
    For j = 1 To NumDataPoints(i)
        If DistTable(j, i) > CriteriaTable(CriteriaRow + 2, i) Then
            CriteriaTable(CriteriaRow + 2, i) = DistTable(j - 1, i)
            Exit For
        ElseIf j = NumDataPoints(i) Then
            CriteriaTable(CriteriaRow + 2, i) = DistTable(j, i)
            Exit For
        End If
    Next j
Next i

' Reset all increments and iteration counts before
' starting to evaluate the next row.

For i = 1 To NumChem
    Increment(i) = (MaxConc(i) - MinConc(i)) / NumIncrements
Next i

Next CriteriaRow

Call CountAETs

End Sub

```

This is the end of FPMCalc.xls mod08FloatingPercentilePass2.
--

3.5.10 mod09CountCopyData

This module contains 9 macros:

Count determines the number of different analytes, data points, etc., that will be needed in later routines.

TestRowCount sets the TestCountStatus to "True" when the CountTrueFalse macro is called directly from the "Count Hits/NoHits in Test Row" button on the [Criteria] page.

CountTrueFalse compares analyte concentrations listed in the Test Row on the [Criteria] page to the current data set and counts the number of true and false positives and negatives that would result if the numbers in the test row were the sediment standards

CountFalsePos(Delta) counts the number of false positives that result from using the data in the specified row on the Criteria Table for the screening values.

CountFalseNeg counts the number of false negatives and calculates the %FN that results from using the data in the specified row on the Criteria Table for the screening values.

CountAETs counts the number of chemicals that are AET at all selected levels, marks the AETs for purposes of sorting, and sorts the Criteria Table.

CopyChemData copies the data table from the FPMAnova spreadsheet and pastes it into this FPMCCalc spreadsheet.

ShowReliabilityDefs displays the ReliabilityDefs form and is activated by a button on the [Criteria] page.

ShowCriteriaDefs displays the CriteriaDefs form and is activated by a button on the [Criteria] page.

The code and comments for this module are shown below.

```
Option Private Module
Option Explicit
```

```
' *****
' Count() determines the number of different analytes, data
' points, etc., that will be needed in later routines.
' *****

Sub Count()

' Update file name just in case user has saved it under new name.

ThisFileName = ActiveWorkbook.Name

' Count the number of elements in some of the data sets used in this procedure.

NumChem = Application.CountA(DTHeader)
NumData = Application.CountA(DataTable)
NumSamples = Application.CountA(DTSample)
NumIndet = Application.CountIf(DTHit, -1)
NumNoHits = Application.CountIf(DTHit, 0)
NumHits = Application.CountIf(DTHit, 1)
```

```
' Sort the raw data in order of station, survey, and sample number in
' preparation for counting the number of different stations present.
```

```
ChemBioData.Sort _
    Key1:=ChemBioData(1, 3), Order1:=xlAscending, _
    Key2:=ChemBioData(1, 2), Order2:=xlAscending, _
    Key3:=ChemBioData(1, 4), Order3:=xlAscending, _
    Header:=xlNo, _
    Orientation:=xlTopToBottom
```

```
End Sub
```

```
' *****
' TestRowCount() sets the TestCountStatus to "True" when the following
' CountTrueFalse macro is called directly from the "Count Hits/NoHits in Test Row"
' button on the [Criteria] page. When the CountTrueFalse macro is called
' from within the FPM procedure the TestCountStatus remains "False".
' The CountTrueFalse macro functions differently depending on where it is used.
' *****
```

```
Sub TestRowCount()
```

```
Call SetRanges
TestCountStatus = "True"
Call CountTrueFalse
```

```
End Sub
```

```
' *****
' CountTrueFalse() compares analyte concentrations listed in the Test Row
' on the [Criteria] page to the current data set and counts the number of
' true and false positives and negatives that would result if the numbers
' in the test row were the sediment standards. This macro is used in two
' places:
'
' (1) It's used to evaluate any data that the user places in the test row.
' In this case it is called by the "Count Hits/NoHits in Test Row" button
' on the [Criteria] page.
'
' (2) It's also used in the counting step of the normal FPM calculation.
' In this case it is called by the FPM macros and works by automatically
' copy the data from the FPM calculation into the test row, running the macro,
' and then putting the test row answers into the appropriate row in the
' main table on the [Criteria] page.
' *****
```

```
Sub CountTrueFalse()
```

```
Dim TestPctFN As Range, TestPctFP As Range, TestPctSens As Range, TestPctEff As Range
Dim TestPctPredHit As Range, TestPctPredNoHit As Range, TestPctRel As Range
```

```
Set TestPctFN = Range("TestPctFN")
Set TestPctFP = Range("TestPctFP")
Set TestPctSens = Range("TestPctSens")
Set TestPctEff = Range("TestPctEff")
Set TestPctPredHit = Range("TestPctPredHit")
Set TestPctPredNoHit = Range("TestPctPredNoHit")
Set TestPctRel = Range("TestPctRel")
```

```
TestTrueHits = 0: TestTrueNoHits = 0
TestFalseHits = 0: TestFalseNoHits = 0
```

```

For i = 1 To NumChem
    If TestMass(i) = "" Then
        TestCount(i) = "-"
    Else
        TestCount(i) = 0
    End If
Next i

'    When used as described in case (1) above, TestCountStatus = "True".
'    When used as described in case (2) above, TestCountStatus = "False".

If TestCountStatus = "True" Then
    Application.ScreenUpdating = False

    For i = 1 To NumChem
        If TestMass(i) = "" Then
            TestTableHeader(i) = "-"
        Else
            TestTableHeader(i) = CriteriaHeader(i)
        End If
    Next i

    For i = 1 To NumSamples
        TestCorrect(i) = 0: TestFalseNeg(i) = 0: TestFalsePos(i) = 0
    Next i

End If

For j = 1 To NumSamples

    If TestCountStatus = "True" Then
        TestHitNoHit(j) = DTHit(j): TestSurvey(j) = DTSurvey(j)
        TestStation(j) = DTStation(j): TestSample(j) = DTSample(j)
    End If

    PredictedHitFlag = 0
    For k = 1 To NumChem
        If (DataTable(j, k) <> "" And TestMass(k) <> "") Then
            If DataTable(j, k) > TestMass(k) Then
                PredictedHitFlag = 1
                Exit For
            End If
        End If
    Next k

    If (PredictedHitFlag = 1 And DTHit(j) = 1) Then
        TestTrueHits = TestTrueHits + 1
        If TestCountStatus = "True" Then TestCorrect(j) = 1

    ElseIf (PredictedHitFlag = 0 And DTHit(j) = 0) Then
        TestTrueNoHits = TestTrueNoHits + 1
        If TestCountStatus = "True" Then TestCorrect(j) = 1

    ElseIf (PredictedHitFlag = 1 And DTHit(j) = 0) Then
        TestFalseHits = TestFalseHits + 1

    ElseIf (PredictedHitFlag = 0 And DTHit(j) = 1) Then
        TestFalseNoHits = TestFalseNoHits + 1
        If TestCountStatus = "True" Then TestFalseNeg(j) = 1
    End If

Next j

```



```

' Count the number of false positives for each chemical at each station

For j = 1 To NumSamples
    For k = 1 To NumChem
        If DTHit(j) <> 0 Then Exit For
        If (DataTable(j, k) <> "" And TestMass(k) <> "") Then
            If DataTable(j, k) > TestMass(k) Then
                TestCount(k) = TestCount(k) + 1
                TestFalsePos(j) = TestFalsePos(j) + 1
                TestDataTable(j, k) = "FP"
            End If
        End If
    Next k
Next j

TestPctFN = 100 * TestFalseNoHits / (TestFalseNoHits + TestTrueHits)
TestPctFP = 100 * TestFalseHits / (TestFalseHits + TestTrueNoHits)
TestPctSens = 100 * TestTrueHits / (TestFalseNoHits + TestTrueHits)
TestPctEff = 100 * TestTrueNoHits / (TestFalseHits + TestTrueNoHits)
TestPctPredHit = 100 * TestTrueHits / (TestTrueHits + TestFalseHits)
TestPctPredNoHit = 100 * TestTrueNoHits / (TestTrueNoHits + TestFalseNoHits)
TestPctRel = 100 * (TestTrueHits + TestTrueNoHits) / (TestTrueHits + TestTrueNoHits +
TestFalseHits + TestFalseNoHits)

If TestCountStatus = "True" Then
    Application.ScreenUpdating = True
    TestCountStatus = "False"
End If

End Sub

' *****
' CountFalsePos(Delta) counts the number of false positives that result from using the data
' in the specified row on the Criteria Table for the screening values.
' *****

Sub CountFalsePos(Delta)

' When "Delta" = 0, "Criteria + Delta" is the initial concentration and
' "Criteria + Delta + 1" (i.e., CriteriaRow + 1) is the row that stores the original #FPs.
' When "Delta" = 2, "Criteria + Delta" is the current concentration being tested and
' "Criteria + Delta + 1" (i.e., CriteriaRow + 3) is the row that stores the current #FPs.

For j = 1 To NumChem: CriteriaTable(CriteriaRow + Delta + 1, j) = 0: Next j

For j = 1 To NumSamples
    For k = 1 To NumChem
        If DTHit(j) <> 0 Then Exit For 'Bio test must be "No Hit" or sample cannot be a FP.
        If DataTable(j, k) <> "" Then
            If DataTable(j, k) > CriteriaTable(CriteriaRow + Delta, k) Then
                CriteriaTable(CriteriaRow + Delta + 1, k) = CriteriaTable(CriteriaRow + Delta
+ 1, k) + 1
            End If
        End If
    Next k
Next j

End Sub

```

```

' *****
' CountFalseNeg() counts the number of false negatives and calculates
' the %FN that results from using the data in the specified
' row on the Criteria Table for the screening values.
' *****

Sub CountFalseNeg()

NewFalseNoHits = 0

For j = 1 To NumSamples
    For k = 1 To NumChem

        If DTHit(j) = 0 Then
            PredictedNoHitFlag = 1      'If Bio test is "No Hit" then sample cannot be a FN.
            Exit For
        Else: PredictedNoHitFlag = 0
        End If

        If DataTable(j, k) <> "" Then
            If DataTable(j, k) > CriteriaTable(CriteriaRow + 2, k) Then
                PredictedNoHitFlag = 1
                Exit For
            End If
        End If

    Next k
    If PredictedNoHitFlag = 0 Then NewFalseNoHits = NewFalseNoHits + 1
Next j

NewPercentFalseNegatives = 100 * NewFalseNoHits / NumHits

End Sub

' *****
' CountAETs() counts the number of chemicals that are AET at all selected levels,
' marks the AETs for purposes of sorting, and sorts the Criteria Table.
' *****

Sub CountAETs()

NumAETChem = 0

If ScreenAETs = "Y" Then
    For i = 1 To NumChem
        Range(CriteriaTable(1, i), CriteriaTable(50, i)).Select
        Set WorkRange = Selection
        If Application.CountIf(WorkRange, "AET") = NumSteps Then
            NumAETChem = NumAETChem + 1
        End If
    Next i
End If

NumFPMChem = NumChem - NumAETChem

End Sub

```

```

' *****
' CopyChemData() Copies the data table from the FPMAnova
' spreadsheet and pastes it into this FPMCals spreadsheet.
' *****

Sub CopyChemData()

Range("A1").Select
Application.ScreenUpdating = False

Call SetRanges
ThisFileName = ActiveWorkbook.Name

SourceFile = AnovaFileName
DestinationFile = ThisFileName

On Error GoTo EndThis

Windows(SourceFile).Activate
Range("FullFinalDataTable").Copy

Windows(DestinationFile).Activate
Sheets("DataTable").Select
Range("FullDataTable").PasteSpecial Paste:=xlPasteValues
Application.CutCopyMode = False

For i = 1 To 100
    DTHeader(i).Select
    With Selection
        .Columns.AutoFit
        If .ColumnWidth < 10 Then .ColumnWidth = 10
    End With
Next i

Range("A1").Select

Application.ScreenUpdating = True
Exit Sub

EndThis:

MsgBox ("          'Copy Anova Data' did not work." & Chr(13) & _
"      Make sure that the source file is open" & Chr(13) & _
"and in the same folder as this spreadsheet," & Chr(13) & _
"      or check the spelling and try again.")

End

End Sub

' *****
' ShowReliabilityDefs() displays the ReliabilityDefs form
' and is activated by a button on the [Criteria] page.
' *****

Sub ShowReliabilityDefs()
    Load ReliabilityDefs
    ReliabilityDefs.Show
End Sub

```

```

' *****
' ShowCriteriaDefs() displays the CriteriaDefs form and is
' activated by a button on the [Criteria] page.
' *****

Sub ShowCriteriaDefs()
    Load CriteriaDefs
    CriteriaDefs.Show
End Sub

```

This is the end of FPMCalc.xls mod09CountCopyData.
--

3.5.11 mod10FormatData

This module contains 6 macros:

Auto_Open sets up the appearance of the workbook upon opening.

SetUpScreen closes toolbars and sets up the worksheets in preparation for using the program. It's called from the Auto_Open procedure.

CenterText centers selected text and adjusts columns to appropriate widths.

DataFormat formats the data on the [Criteria] and [DataStorage] pages.

ShowHideCriteriaRows either shows or hides the extra rows of data on the [Criteria] page. It's activated by the "Show Additional Calculation Details" button on that page.

Auto_Close restores original toolbars, menus, etc. It works automatically on closing.

The code and comments for this module are shown below.

```
Option Private Module
Option Explicit
```

```
' *****
' Auto_Open() sets up the appearance of the workbook upon opening.
' *****
```

```
Sub Auto_Open()
```

```
Application.ScreenUpdating = False
Application.StatusBar = ""
```

```
Call SetRanges
Call SetUpScreen
Call Count
```

```
Sheets("ControlScreen").Activate
Application.ScreenUpdating = True
```

```
End Sub
```

```
' *****
' SetUpScreen() closes toolbars and sets up the worksheets
' in preparation for using the program. It's called from
' the Auto_Open procedure.
' *****
```

```
Sub SetUpScreen()
```

```
Dim Bar As Object, BarCount As Integer
```

```
' This sets each worksheet back to a standard top-left
' orientation with the cursor hidden off screen and hides
' the sheet that's used to store reference information.
```

```

For i = 8 To 1 Step -1
    Sheets(i).Visible = True
    Sheets(i).Activate
    ActiveWindow.DisplayHeadings = False
    Application.DisplayCommentIndicator = xlCommentIndicatorOnly
    ActiveWindow.ScrollColumn = 1
    ActiveWindow.ScrollRow = 1
    Range("A1").Select
Next i

Sheets(8).Visible = False

' To create more space, hide the Formula Bar and the Comment Indicator

Application.DisplayCommentIndicator = False
Application.DisplayFormulaBar = False

' Keep track of what toolbars are open so they can be reopened
' before exiting. Names are stored on the hidden worksheet [Names].
' Then hide the open toolbars to create more space on the screen.

ToolBarStorage.ClearContents
BarCount = 0

For Each Bar In Toolbars
    If Bar.Visible = True Then
        BarCount = BarCount + 1
        ToolBarStorage(BarCount) = Bar.Name
        Bar.Visible = False
    End If
Next Bar

End Sub

' *****
' CenterText() centers selected text and adjusts columns
' to appropriate widths.
' *****

Sub CenterText()

With Selection
    .HorizontalAlignment = xlCenter
    .Columns.AutoFit
End With

End Sub

' *****
' DataFormat() formats the data on the [Criteria] and
' [DataStorage] pages.
' *****

Sub DataFormat()

Sheets("Criteria").Select

Range("CriteriaHeader").Select
Call CenterText

```

```

For i = 1 To 100
    CriteriaHeader(i).Select
    If CriteriaHeader(i) = "" Then
        Selection.ColumnWidth = 10
    Else
        If Selection.ColumnWidth < 10 Then Selection.ColumnWidth = 10
    End If
Next i

Sheets("DataStorage").Select

Range("StorageHeader").Select
Call CenterText

For i = 1 To 100
    StorageHeader(i).Select
    If StorageHeader(i) = 0 Then
        Selection.ColumnWidth = 10
    Else
        If Selection.ColumnWidth < 10 Then Selection.ColumnWidth = 10
    End If
Next i

End Sub

' *****
' ShowHideCriteriaRows() either shows or hides the extra rows
' of data on the [Criteria] page. It's activated by the
' "Show Additional Calculation Details" button on that page.
' *****

Sub ShowHideCriteriaRows()

Dim ShowHide As Range
Set ShowHide = Range("ShowHide")

Call SetRanges

If ShowHide = False Then
    ShowHide = True
    ActiveSheet.Shapes("Button 93").Select
    Selection.Characters.Text = "Show Additional Calculation Details"
    Range("A1").Select
    Range(CriteriaTable(1, 1), CriteriaTable(2, 1)).EntireRow.Hidden = True
    Range(CriteriaTable(4, 1), CriteriaTable(7, 1)).EntireRow.Hidden = True
    Range(CriteriaTable(9, 1), CriteriaTable(12, 1)).EntireRow.Hidden = True
    Range(CriteriaTable(14, 1), CriteriaTable(17, 1)).EntireRow.Hidden = True
    Range(CriteriaTable(19, 1), CriteriaTable(22, 1)).EntireRow.Hidden = True
    Range(CriteriaTable(24, 1), CriteriaTable(27, 1)).EntireRow.Hidden = True
    Range(CriteriaTable(29, 1), CriteriaTable(32, 1)).EntireRow.Hidden = True
    Range(CriteriaTable(34, 1), CriteriaTable(37, 1)).EntireRow.Hidden = True
    Range(CriteriaTable(39, 1), CriteriaTable(42, 1)).EntireRow.Hidden = True
    Range(CriteriaTable(44, 1), CriteriaTable(47, 1)).EntireRow.Hidden = True
    Range(CriteriaTable(49, 1), CriteriaTable(50, 1)).EntireRow.Hidden = True
ElseIf ShowHide = True Then
    ShowHide = False
    ActiveSheet.Shapes("Button 93").Select
    Selection.Characters.Text = "Hide Additional Calculation Details"
    Range("A1").Select
    Range(CriteriaTable(1, 1), CriteriaTable(50, 1)).EntireRow.Hidden = False
End If

End Sub

```

```
' *****
' Auto_Close() restores original toolbars, menus, etc.
' Works automatically on closing.
' *****
```

```
Sub Auto_Close()

    Call SetRanges
    On Error Resume Next
    Application.ScreenUpdating = False
    Sheets("ControlScreen").Select

    For i = 1 To Application.CountA(ToolBarStorage)
        Toolbars(ToolBarStorage(i).Value).Visible = True
    Next i

    Application.DisplayFormulaBar = True
    Application.DisplayCommentIndicator = True

End Sub
```

This is the end of FPMCalc.xls mod10FormatData.

This is the end of the macro code for the FPMCalc.xls Worksheet.
--